МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ – МСХА имени К.А. ТИМИРЯЗЕВА»

(ФГБОУ ВО РГАУ - МСХА имени К.А. Тимирязева)

Институт экономики и управления АПК
Кафедра статистики и кибернетики

УТВЕРЖДАЮ:
Директор института
экономики и управления АПК
_____ Л.И. Хоружий
«28» августа 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ
Б1.В.16 Фронтенд и бэкенд разработка

для подготовки бакалавров

ФГОС ВО

Направление: 09.03.02 «Информационные системы и технологии»
Направленность:
«Фуллстек разработка»

Курс 3
Семестр 5,6

Форма обучения: очная

Год начала подготовки: 2025

Москва, 2025

Разработчики:
Демичев В.В., канд. экон. наук, доцент _____ «26» августа 2025 г.
(ФИО, ученая степень, ученое звание) (подпись)

Храмов Д.Э., ассистент _____ «26» августа 2025 г.
(ФИО, ученая степень, ученое звание) (подпись)

Рецензент:

Вахрушева И.А., канд. пед. наук _____ «26» августа 2025 г.
(ФИО, ученая степень, ученое звание) (подпись)

Программа составлена в соответствии с требованиями ФГОС ВО по направлению подготовки 09.03.02 Информационные системы и технологии, профессионального стандарта и учебного плана 2025 года начала подготовки.

Программа обсуждена на заседании кафедры статистики и кибернетики. Протокол №11 от «26» августа 2025 г.

И. о. зав. кафедрой Уколова А.В., канд. экон. наук, доцент _____
(ФИО, ученая степень, ученое звание) (подпись)

«26» августа 2025 г.

Согласовано:
Председатель учебно-методической
комиссии института экономики и управления АПК
Гупалова Т.Н., канд. экон. наук, доцент _____
(ФИО, ученая степень, ученое звание) (подпись)

протокол №1 «28» августа 2025 г.

И. о. зав. выпускающей кафедрой
статистики и кибернетики
Уколова А.В., канд. экон. наук, доцент _____
(ФИО, ученая степень, ученое звание) (подпись)

«28» августа 2025 г.

Заведующий отделом комплектования ЦНБ _____
(подпись)

СОДЕРЖАНИЕ

АННОТАЦИЯ

рабочей программы учебной дисциплины Б1.В.16 «Фронтенд и бэкенд разработка» для подготовки бакалавров по направлению 09.03.02 Информационные системы и технологии, направленности «Фуллстек разработка»

Цель освоения дисциплины: формирование у студентов профессиональных компетенций в области проектирования, реализации и развёртывания современных fullstack-веб-приложений с использованием актуальных JavaScript/TypeScript-технологий и архитектурных подходов. В ходе изучения дисциплины студенты осваивают методы разработки как клиентской части на основе компонентного подхода (React), так и серверной логики с использованием Node.js и RESTful-архитектуры, а также приобретают навыки проектирования API, работы с нереляционными базами данных, реализации аутентификации на основе JWT, обеспечения взаимодействия фронтенда и бэкенда, а также подготовки fullstack-проектов к промышленной эксплуатации и развёртыванию в облачных средах.

Место дисциплины в учебном плане: дисциплина включена в часть, формируемую участниками образовательных отношений дисциплин учебного плана по направлению подготовки 09.03.02 Информационные системы и технологии.

Требования к результатам освоения дисциплины: в результате освоения дисциплины формируются следующие компетенции (индикаторы): ПКос-3(ПКос-3.1, ПКос-3.2, ПКос-3.3), ПКос-4(ПКос-4.1, ПКос-4.2, ПКос-4.3), ПКос-5(ПКос-5.1, ПКос-5.2, ПКос-5.3), ПКос-6(ПКос-6.1,ПКос-6.2, ПКос-6.3), ПКос-7(ПКос-7.1, ПКос-7.2, ПКос-7.3), ПКос-8(ПКос-8.1, ПКос-8.2, ПКос-8.3).

Краткое содержание дисциплины:

Архитектурные паттерны fullstack-приложений: клиент-сервер, REST, слоистая архитектура (controllers, services, models). Обзор современных JavaScript/TypeScript-технологий: React, Node.js, Express — их назначение, сильные стороны и типичные сценарии применения. Основы языка TypeScript: типизация, интерфейсы, асинхронность, модули. Разработка клиентской части: компонентный подход, хуки (useState, useEffect, useContext), маршрутизация (React Router), управление состоянием и запросами к API (fetch, axios, React Query). Разработка серверной части: создание RESTful-сервисов на Express, middleware, обработка JSON, валидация входных данных. Работа с нереляционными базами данных: MongoDB, моделирование данных, CRUD-операции, агрегация с использованием Mongoose. Аутентификация и авторизация: JWT, refresh-токены, защищённые маршруты, хранение токенов. Управление статическими и медиа-файлами. Обработка ошибок и логирование на клиенте и сервере. Тестирование fullstack-приложений: unit-тесты (Jest), компонентные тесты (React Testing Library), интеграционные тесты (Postman). Подготовка к развёртыванию: сборка фронтенда (Vite), настройка production-окружения для Node.js, управление переменными окружения, контейнеризация с Docker, публикация в облачных средах (Vercel, Render, Railway).

Общая трудоемкость дисциплины: 288 / 8 (часы/зач. ед.)

Промежуточный контроль: зачет, экзамен

1. Цель освоения дисциплины

Целью освоения дисциплины «Фронтенд и бэкенд разработка» является овладение студентами знаний архитектурных принципов построения fullstack-веб-приложений, а также особенностей клиент-серверного взаимодействия, основных подходов к проектированию RESTful API и организации компонентной архитектуры на стороне клиента. Студент должен знать принципы маршрутизации на фронтенде и бэкенде, обработки HTTP-запросов и ответов, валидации данных, реализации аутентификации на основе JWT, работы с нереляционными базами данных и управления состоянием приложения; понимать особенности типизированной разработки на TypeScript, механизмы управления зависимостями, обработки ошибок, кэширования запросов и защиты от уязвимостей (XSS, инъекции в MongoDB, подделка токенов); а также знать практики подготовки fullstack-приложений к развёртыванию и эксплуатации в production-средах. По окончанию изучения дисциплины студент должен уметь проектировать структуру fullstack-приложения в соответствии с поставленной задачей, создавать клиентские компоненты и маршруты на React, разрабатывать RESTful-сервисы на Express, реализовывать обработку пользовательского ввода через формы и API-запросы, интегрировать работу с MongoDB, реализовывать механизмы разграничения доступа, обрабатывать загрузку файлов, управлять аутентификационными токенами, а также настраивать сборку фронтенда и бэкенда и готовить приложение к развёртыванию в облачных средах. Также по окончанию изучения дисциплины студент должен владеть навыками разработки полнофункциональных веб-приложений с использованием компонентного подхода в React, навыками создания легковесных и масштабируемых серверных API на Node.js и Express, методами тестирования клиентской и серверной логики, техниками обеспечения безопасности fullstack-приложений, а также практиками документирования API (с использованием Swagger/OpenAPI) и оформления архитектурных решений в соответствии с профессиональными стандартами современной веб-разработки.

2. Место дисциплины в учебном процессе

Дисциплина «Фронтенд и бэкенд разработка» включена в часть, формируемую участниками образовательных отношений дисциплин учебного плана. Дисциплина «Фронтенд и бэкенд разработка» реализуется в соответствии с требованиями ФГОС ВО, ОПОП ВО и Учебного плана по направлению 09.03.02 Информационные системы и технологии.

Дисциплина «Фронтенд и бэкенд разработка» изучается на третьем курсе образовательного цикла.

Предшествующими курсами, включенными в учебный план, на которых непосредственно базируются дисциплина «Фронтенд и бэкенд разработка», являются «Алгоритмизация и программирование», «Веб-разработка», «Теория информации».

Особенностью дисциплины является фокус на практико-ориентированном освоении современного JavaScript/TypeScript-стека fullstack-разработки, включая фронтенд-фреймворк React и бэкенд-платформу Node.js с

Express, с акцентом на проектирование архитектуры приложений, организацию взаимодействия клиента и сервера, реализацию аутентификации, работу с нереляционными базами данных, обеспечение безопасности и подготовку fullstack-проектов к развёртыванию в облачных средах.

Рабочая программа дисциплины «Фронтенд и бэкенд разработка» для инвалидов и лиц с ограниченными возможностями здоровья разрабатывается индивидуально с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся.

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

Образовательные результаты освоения дисциплины обучающимся, представлены в таблице 1.

Таблица 1

Требования к результатам освоения учебной дисциплины «Фронтенд и бэкенд разработка»

№ п/п	Код компетенции	Содержание компетенции (или её части)	Индикаторы компетенций	В результате изучения учебной дисциплины обучающиеся должны:		
				знать	уметь	владеть
1.	ПКос-3	Способен проектировать и разрабатывать информационные ресурсы с использованием полного стека технологий	ПКос-3.1	Студент должен знать: архитектурные принципы fullstack-приложений: разделение на клиент и сервер, REST-взаимодействие, слоистую архитектуру (controllers, services, models); особенности TypeScript – типизация, интерфейсы, асинхронность; структуру React-приложения, компоненты, props, state, хуки, маршрутизацию; типовые решения Express; подходы к проектированию схем в MongoDB и принципы работы с Mongoose.		
			ПКос-3.2		Студент должен уметь: проектировать структуру fullstack-приложения, выделять экраны, API-эндпоинты, модели данных; использовать типовые шаблоны, компоненты React, роуты Express, CRUD-	

					операции в MongoDB; применять методы проектирования; интегрировать фронтенд и бэкенд через HTTP-запросы; организовывать проект в соответствии с best practices: папки по функционалу, выделение слоёв, повторное использование кода.	
			ПКос-3.3			Студент должен владеть: навыками проектирования и разработки полноценных fullstack-приложений; практиками архитектурного проектирования, выбор между локальным состоянием и запросами к API, организация аутентификации, работа с ошибками; навыками проектирования схем данных в MongoDB с учётом запросов; техниками рефакторинга: выделение хуков, сервисов, DTO.
2.	ПКос-4	Способен осуществлять разработку, отладку и рефакторинг кода программного обеспечения, интеграцию программных модулей и компонент, в том числе взаимодействующих с внешней средой, средствами выбранных язы-	ПКос-4.1	Студент должен знать: синтаксис и особенности TypeScript, типы, интерфейсы, enum, async/await, generics; стандартные библиотеки и инструменты: React (хуки, контекст), Express (middleware, роутеры), Axios/Fetch;		

		ков программирования		среды разработки: VS Code, DevTools, Postman; типы ошибок: синтаксические, runtime, сетевые – и их форматы в консоли и логах; методы повышения читаемости, именование, комментирование, разделение ответственностей.		
			ПКос-4.2		Студент должен уметь: писать чистый, типизированный код на TypeScript для фронтенда и бэкенда; использовать VS Code для навигации, отладки, рефакторинга; интерпретировать ошибки: отличать ошибки CORS от 404, понимать stack trace; применять инструменты ESLint, Prettier, TypeScript compiler; выявлять и исправлять баги.	
			ПКос-4.3			Студент должен владеть: навыками разработки, отладки и рефакторинга fullstack-кода; техниками оптимизации, мемоизация компонентов, дебаунс запросов, ленивая загрузка; практиками анализа кода: code review, статический анализ; навыками отладки на уровне взаимодействия: проверка запросов

						в DevTools, логирование на сервере, сопоставление типов фронт/бэк.
3.	ПКос-5	Способен осуществлять методическое сопровождение испытаний системы	ПКос-5.1	Студент должен знать: критерии юзабилити для интерфейсов, скорость отклика, предсказуемость навигации, обратная связь, доступность; методы юзабилити-тестирования; стандарты Material Design и их применимость к вебу; инструменты анализа Lighthouse, axe DevTools.		
			ПКос-5.2		Студент должен уметь: разрабатывать пользовательские сценарии; проводить экспертную оценку интерфейса по чек-листу; использовать Lighthouse для оценки производительности и доступности; фиксировать замечания в виде отчёта с рекомендациями..	
			ПКос-5.3			Студент должен владеть: навыками анализа полноты покрытия пользовательских сценариев в интерфейсе; методикой выбора регистрируемых параметров: время выполнения задачи, количество оши-

						бок, субъективная оценка; практиками итеративного улучшения интерфейса на основе обратной связи.
4.	ПКос-6	Способен организовать работы по обеспечению безопасности информационных ресурсов	ПКос-6.1	Студент должен знать: основные угрозы безопасности веб-приложений: XSS, CSRF, инъекции в MongoDB, подделка JWT-токенов; принципы безопасной аутентификации: хранение токенов, refresh-токены, rate limiting; настройки Express: helmet, cors, rate-limit; основы HTTPS и CSP.		
			ПКос-6.2		Студент должен уметь: настраивать middleware безопасности: helmet, cors, express-rate-limit; валидировать входные данные на бэкенде; защищать маршруты с помощью JWT; анализировать логи на предмет подозрительной активности; документировать регламенты безопасности (где хранятся ключи, как обновляются токены и пр.).	
			ПКос-6.3			Студент должен владеть: навыками администрирования

						базовой безопасности fullstack-приложения; практиками безопасной разработки; навыками настройки production-окружения с учётом безопасности.
5.	ПКос-7	Способен осуществлять концептуально-логическое проектирование системы, разрабатывать техническое задание	ПКос-7.1	Студент должен знать: особенности предметной области АПК: учёт полей, техники, урожайности и соответствующие пользовательские роли; основы REST API, статусы, форматы, версионирование; платформы: React (фронт), Node.js/Express (бэк), MongoDB (БД); принципы работы с внешними сервисами: карты, погода, IoT-датчики.		
			ПКос-7.2		Студент должен уметь: разрабатывать тест-планы для интеграционного тестирования, проверять корректности запросов, обработки ошибок сети, валидации; готовить тестовые данные; интерпретировать бизнес-требования и переводить их в тестовые сценарии; работать в команде: согласо-	

					вывать API-контракт, сообщать об ошибках.	
			ПКос-7.3			Студент должен владеть: навыками разработки стратегии интеграционного тестирования, определять критические точки, выбирать инструменты, проводить приоритизацию сценариев; практиками управления процессом тестирования: ведение чек-листов, отслеживание дефектов, верификация исправлений.
6.	ПКос-8	Способен проводить анализ и формализацию требований к информационным ресурсам	ПКос-8.1	Студент должен знать: архитектуру веб-приложений, клиент-сервер, HTTP/HTTPS, REST; сетевые протоколы DNS, TCP, TLS; основы MongoDB: документы, коллекции, индексы; стандарты взаимодействия JSON, OpenAPI/Swagger; методики моделирования: пользовательские сценарии, диаграммы потоков данных.		
			ПКос-8.2		Студент должен уметь: анализировать требования; обосновывать выбор: React vs Vue, MongoDB vs	

					PostgreSQL; применять формализацию: написать спецификацию API в формате OpenAPI; использовать инструменты Swagger Editor, draw.io для диаграмм	
			ПКос-8.3			Студент должен владеть: навыками составления формализованных описаний, техническое задание, API-документация, схемы данных; практиками разработки алгоритмов от сценария к последовательности шагов фронт/бэк; навыками оформления решений в соответствии с профессиональными стандартами.

4. Структура и содержание дисциплины

4.1 Распределение трудоёмкости дисциплины по видам работ по семестрам

Общая трудоёмкость дисциплины составляет 8 зач.ед. (288 часов), их распределение по видам работ и семестрам представлено в таблице 2.

Таблица 2

Распределение трудоёмкости дисциплины по видам работ

Вид учебной работы	**Трудоёмкость**		
	час. всего	**в т.ч. по семестрам**	
		№ 5	**№ 6**
Общая трудоёмкость дисциплины по учебному плану	**288**	**108**	**180**
1. Контактная работа:	**105,65**	**53,25**	**52,4**
Аудиторная работа	**105,65**	**53,25**	**52,4**
лекции (Л)	**32**	16	16
практические занятия (ПЗ)	**68**	34	34
курсовое проектирование (КП)	**3**	3	-
консультации перед экзаменом	**2**	-	2
контактная работа на промежуточном контроле (КРА)	**0,65**	0,25	0,4
2. Самостоятельная работа (СРС)	**155,35**	**54,75**	**100,6**
самостоятельное изучение разделов, самоподготовка (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к практическим занятиям)	**155,35**	54,75	100,6
Подготовка к экзамену (контроль)	**27**	-	27
Вид промежуточного контроля:	Зачет, Экзамен		

4.2 Содержание дисциплины

Таблица 3

Тематический план учебной дисциплины

Наименование разделов и тем дисциплин (укрупнённо)	**Всего**	**Аудиторная работа**			**Внеаудиторная работа СР**
		Л	**ПЗ всего**	**ПКР**	
Тема 1 «Введение в fullstack-разработку и экосистему JavaScript/TypeScript»	4,75	1	-	-	3,75
Тема 2 «Основы TypeScript»	13	3	4	-	6
Тема 3 «Фронтенд: основы React»	14	2	6	-	6
Тема 4 «Фронтенд: хуки и жизненный цикл»	15	2	6	-	7
Тема 5 «Фронтенд: маршрутизация и навигация»	11	1	4	-	6
Тема 6 «Бэкенд: основы Node.js и Express»	13	2	4	-	7
Тема 7 «База данных: MongoDB и	13	2	4	-	7

Наименование разделов и тем дисциплин (укрупнённо)	**Всего**	**Аудиторная работа**			**Внеаудиторная работа СР**
		Л	**ПЗ всего**	**ПКР**	
Mongoose»					
Тема 8 «Аутентификация и авторизация»	13	2	4	-	7
Тема 9 «Взаимодействие фронтенда и бэкенда»	8	1	2	-	5
Курсовой проект (КП) (консультация, защита)	3	-	-	3	-
Контактная работа на промежуточном контроле	0,25	-	-	0,25	-
Всего за 5 семестр	**108**	**16**	**34**	**3,25**	**54,75**
Тема 10 «Архитектура fullstack-приложений»	18	2	4	-	12
Тема 11 «Современный React»	17	2	4	-	11
Тема 12 «Валидация и обработка ошибок»	16	2	4	-	10
Тема 13 «Тестирование fullstack-приложений»	26	2	6	-	18
Тема 14 «Работа с файлами и медиа»	11	1	2	-	8
Тема 15 «DevOps и инструменты разработчика»	21	2	5	-	14
Тема 16 «Деплой и облачные среды»	18	2	4	-	12
Тема 17 «Безопасность fullstack-приложений»	17	2	4	-	11
Тема 18 «Документирование и профессиональные стандарты»	6,6	1	1	-	4,6
контактная работа на промежуточном контроле (КРА)	0,4	-	-	0,4	-
Подготовка к экзамену (контроль)	27	-	-	-	27
Консультации перед экзаменом	2	-	-	2	-
Всего за 6 семестр	**180**	**16**	**34**	**2,4**	**127,6**
Итого по дисциплине	**288**	**32**	**68**	**5,65**	**182,35**

Введение в fullstack-разработку и экосистему JavaScript/TypeScript.

Сравнение современных fullstack-подходов: MERN, MEAN, Python-стек – их сильные стороны, ограничения и типичные сценарии применения. Обзор экосистемы JavaScript/TypeScript: npm, yarn, pnpm, структура проекта, управление зависимостями, выбор стека под задачу. Основы клиент-серверной архитектуры, роль TypeScript в повышении надёжности кода, настройка среды разработки (VS Code, Git, Node.js).

Основы TypeScript.

Синтаксис и особенности TypeScript: строгая типизация, примитивные и составные типы, интерфейсы, типы объектов, объединения и пересечения. Работа с функциями: типизация параметров и возвращаемых значений, необязательные и остаточные параметры. Асинхронность: Promise, async/await. Модульная система, компиляция, режим strict, интеграция с JavaScript-библиотеками.

Фронтенд: основы React.

Компонентный подход в React: функциональные компоненты, JSX, props, состояние через useState. Жизненный цикл компонента: монтирование, обновление, размонтирование. Обработка событий, условный рендеринг, списки и ключи. Основы компоновки: вложение компонентов, передача данных сверху вниз. Принципы реактивного UI и иммутабельности.

Фронтенд: хуки и жизненный цикл.

Основные хуки React: useState для управления состоянием, useEffect для побочных эффектов (загрузка данных, подписки). Правила хуков, зависимости useEffect. Дополнительные хуки: useCallback, useMemo – для оптимизации. Кастомные хуки как способ переиспользования логики. Обработка жизненного цикла: загрузка, обновление, очистка ресурсов.

Фронтенд: маршрутизация и навигация.

Организация многостраничных приложений с помощью React Router: BrowserRouter, Routes, Route, Link, NavLink. Динамические маршруты с параметрами, вложенные маршруты. Программная навигация через useNavigate. Активные ссылки, защита маршрутов (предварительная), интеграция навигации с состоянием приложения.

Бэкенд: основы Node.js и Express.

Архитектура серверных приложений на Node.js: событийный цикл, неблокирующий ввод-вывод. Фреймворк Express: создание сервера, маршрутизация (GET, POST, PUT, DELETE), middleware, обработка JSON. Принципы REST: ресурсы, HTTP-методы, статусы. Обработка CORS, логирование, централизованная обработка ошибок.

База данных: MongoDB и Mongoose.

Нереляционные базы данных: особенности MongoDB, документы, коллекции, ObjectId. Моделирование данных для fullstack-приложений. ODM Mongoose: схемы, модели, валидация, CRUD-операции. Связи между документами (вложение и ссылки). Агрегация, индексы, подключение к MongoDB Atlas.

Аутентификация и авторизация.

Принципы аутентификации в веб-приложениях: сессии vs токены. Реализация JWT (JSON Web Token): генерация, верификация, refresh-токены. Middleware авторизации в Express. Безопасное хранение токенов на фронтенде. Защита маршрутов, роли пользователей (базовая реализация). Обработка ошибок аутентификации.

Взаимодействие фронтенда и бэкенда.

Организация клиент-серверного взаимодействия: HTTP-запросы с использованием fetch и axios. Передача данных в форматах JSON, обработка заголовков. Асинхронная загрузка состояния на фронтенде, индикаторы загрузки, обработка ошибок. Интеграция API-слоя в React-компоненты. Тестирование взаимодействия в режиме разработки.

Архитектура fullstack-приложений.

Слоистая архитектура fullstack-приложений: разделение на контроллеры, сервисы, модели и DTO. Принципы чистой архитектуры и SOLID в контексте JavaScript/TypeScript. Организация проекта: структура папок, модульность, повторное использование кода. Сравнение подходов: монолит vs микросервисы (в

упрощённом виде). Выбор архитектурного стиля под задачу, управление зависимостями, инверсия управления (IoC) через фабрики и инъекцию.

Современный React.

Современные паттерны разработки на React: useContext и useReducer для глобального состояния, кастомные хуки для выделения логики. Управление состоянием с помощью React Query и SWR: кэширование, фоновая синхронизация, обновление данных. Оптимизация производительности: React.memo, useCallback, useMemo, lazy loading и Suspense. Архитектура компонентов: атомарный дизайн, композиция, пропсы vs контекст.

Валидация и обработка ошибок.

Централизованная валидация данных: Zod и Joi для бэкенда, React Hook Form и Yup для фронтенда. Валидация на уровне API-контракта, обработка ошибок валидации и возврат структурированных ответов. Единая стратегия обработки ошибок: middleware в Express, обработчики на фронтенде, пользовательские сообщения. Логирование ошибок, retry-механизмы, graceful degradation.

Тестирование fullstack-приложений.

Методы тестирования fullstack-приложений: unit-тесты (Jest), компонентные тесты (React Testing Library), интеграционные тесты (Postman, SuperTest). Написание тестов для бизнес-логики, API-эндпоинтов и UI-компонентов. Mock-объекты, шпионы, фикстуры. Покрытие кода (coverage), CI-интеграция тестов, стратегии тестирования (top-down, bottom-up).

Работа с файлами и медиа.

Загрузка и обработка файлов на бэкенде: middleware multer, ограничение размера и типа файлов. Хранение файлов: локальная файловая система, облачные хранилища (базово). Отображение изображений на фронтенде: превью, обработка ошибок загрузки. Работа с медиа: оптимизация, форматы, безопасность (проверка MIME-типов). Интеграция с формами и API.

DevOps и инструменты разработчика.

Современный workflow разработчика: Git-стратегии (feature branches, pull requests), code review. Контейнеризация: Dockerfile для фронтенда и бэкенда, docker-compose для локального запуска. CI/CD: GitHub Actions для автоматической сборки и запуска тестов. Инструменты: ESLint, Prettier, Husky, lint-staged – для поддержания качества кода. Управление версиями зависимостей.

Деплой и облачные среды.

Подготовка приложения к развёртыванию: production-сборка (Vite), оптимизация ресурсов. Облачные платформы: Vercel и Netlify для фронтенда, Render и Railway для бэкенда. Управление переменными окружения, подключение внешних сервисов (MongoDB Atlas). Настройка доменов, HTTPS, кэширование. Мониторинг и логирование в production.

Безопасность fullstack-приложений.

Основные угрозы безопасности: XSS, CSRF (ограничено в React), NoSQL-инъекции, подделка JWT-токенов. Защита на бэкенде: middleware helmet, cors, rate limiting, валидация входных данных. Безопасное хранение токенов: httpOnly cookies, refresh-токены. Content Security Policy (CSP), защита от overposting, аудит зависимостей (npm audit).

Документирование и профессиональные стандарты.

Стандарты оформления кода: именование, комментирование, структура проекта. Инструменты: ESLint, Prettier, editorconfig – для единообразия. Документирование API: OpenAPI/Swagger – спецификация, генерация, интеграция. Пользовательская документация: README.md, руководство по развёртыванию. Профессиональные практики: semantic versioning, CHANGELOG, лицензирование.

4.3 Практические занятия

Таблица 4

Содержание практических занятий и контрольные мероприятия

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
Тема 1. «Введение в fullstack-разработку и экосистему JavaScript/TypeScript»	Лекция 1. «Сравнение fullstack-стеков и основы экосистемы JavaScript»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
Тема 2 «Основы TypeScript»	Лекция 2. «Типы, интерфейсы и основы статической типизации»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 3. «Функции, классы и модули в TypeScript»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 4. «Асинхронность: Promise и async/await»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 1. «Ти-	ПКос-3.1, ПКос-3.2,	Устный	4

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
	пизация данных и асинхронные операции»	ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	опрос	
Тема 3 «Фронтенд: основы React»	Лекция 5. «Компоненты, props и JSX»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 6. «Состояние, обработка событий и условный рендеринг»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 2. «Создание UI-компонентов для приложения учёта техники»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос -	3
	Практическая работа 3. «Реализация форм и интерактивности»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос -	3
Тема 4 «Фронтенд: хуки и жизненный цикл»	Лекция 7. «useState и useEffect: управление состоянием и побочными эффектами»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-	-	1

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
		8.3.		
	Лекция 8. «useCallback, useMemo и кастомные хуки»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 4. «Загрузка данных и управление состоянием в компонентах»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	3
	Практическая работа 5. «Создание кастомных хуков для переиспользования логики»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	3
Тема 5 «Фронтенд: маршрутизация и навигация»	Лекция 9. «Многостраничные приложения с React Router»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 6. «Реализация навигации и динамических маршрутов»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	4
Тема 6 «Бэкенд: основы Node.js и Express»	Лекция 10. «Создание сервера и маршрутизация в Express»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-	-	1

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
		7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.		
	Лекция 11. «Middleware, обработка JSON и CORS»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 7. «Разработка REST API для курсового проекта»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	4
Тема 7 «База данных: MongoDB и Mongoose»	Лекция 12. «Схемы, модели и CRUD-операции в Mongoose»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 13. «Связи между документами и агрегация»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 8. «Интеграция MongoDB в бэкенд и реализация CRUD»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос -	4
Тема 8 «Аутентификация и авторизация»	Лекция 14. «JWT: генерация, верификация и middleware авторизации»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-	-	1

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
		6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.		
	Лекция 15. «Регистрация, вход и защита маршрутов	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 9. «Реализация аутентификации в fullstack-приложении»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	4
Тема 9 «Взаимодействие фронтенда и бэкенда»	Лекция 16. «Интеграция API и обработка ошибок на фронтенде»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 10. «Финальная сборка курсового проекта: фронт + бэк»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	2
Тема 10. «Архитектура fullstack-приложений	Лекция 17. «Слоистая архитектура: контроллеры, сервисы, модели»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 18. «DTO, чистый код и принципы SOLID в TypeScript»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3,	-	1

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
		ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.		
	Практическая работа 11. «Рефакторинг курсового проекта: выделение слоёв и DTO»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	4
Тема 11 «Современный React»	Лекция 19. «useContext, useReducer и управление глобальным состоянием»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 20. «React Query: кэширование, мутации, фоновая синхронизация»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 12. «Миграция состояния на React Query и оптимизация компонентов»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	4
Тема 12 «Валидация и обработка ошибок»	Лекция 21. «Валидация с Zod и Joi на бэкенде»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 22. «Централизован-	ПКос-3.1, ПКос-3.2,	-	1

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
	ная обработка ошибок и пользовательские сообщения»	ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.		
	Практическая работа 13. «Интеграция валидации и единой обработки ошибок в проект»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос -	4
Тема 13 «Тестирование fullstack-приложений»	Лекция 23. «Unit-тесты с Jest и компонентные тесты с React Testing Library»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 24. «Интеграционные тесты и стратегии покрытия»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 14. «Написание unit-тестов для бэкенда»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	3
	Практическая работа 15. «Тестирование UI-компонентов и API-интеграции»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-	Устный опрос	3

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
		8.3.		
Тема 14 «Работа с файлами и медиа»	Лекция 25. «Загрузка файлов с multer и безопасность медиа»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 16. «Реализация загрузки изображений для техники/полей»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	2
Тема 15 «DevOps и инструменты разработчика»	Лекция 26. «Git workflow, Docker и контейнеризация fullstack-приложений»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 27. «CI/CD с GitHub Actions и качество кода (ESLint, Prettier)»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 17. «Контейнеризация проекта с Docker и docker-compose»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	3
	Практическая работа 18. «Настройка CI и форматирования кода»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-	Устный опрос	2

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
		7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.		
Тема 16 «Деплой и облачные среды»	Лекция 28. «Деплой фронтенда на Vercel и бэкенда на Render»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 29. «Управление переменными окружения и подключение MongoDB Atlas»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 19. «Публикация fullstack-приложения в облако»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос -	4
Тема 17 «Безопасность fullstack-приложений»	Лекция 30. «Защита от XSS, NoSQL-инъекций и настройка helmet»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Лекция 31. «Безопасная аутентификация и Content Security Policy»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 20. «Аудит и усиление безопасности курсового проекта»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-	Устный опрос	4

Название раздела, темы	№ и название лекций/ практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов
		6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.		
Тема 18 «Документирование и профессиональные стандарты»	Лекция 32. «OpenAPI, README и стандарты оформления кода»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	-	1
	Практическая работа 21. «Оформление документации и финальная проверка проекта»	ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3.	Устный опрос	1

Таблица 5

Перечень вопросов для самостоятельного изучения дисциплины

№ п/п	Название раздела, темы	Перечень рассматриваемых вопросов для самостоятельного изучения
1.	Тема 1. «Введение в fullstack-разработку и экосистему JavaScript/TypeScript»	Сравнение стеков MERN, MEAN и Python-стека: сильные стороны и типичные сценарии применения. Роль TypeScript в повышении надёжности кода по сравнению с JavaScript. Назначение npm, yarn и pnpm, различия и выбор под проект. Как устроена структура типичного fullstack-проекта на React + Express. (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
2.	Тема 2 «Основы TypeScript»	В чём разница между interface и type в TypeScript? Как работают union и intersection types? Почему strict mode рекомендуется включать в production-проектах? Как правильно типизировать асинхронную функцию, возвращающую Promise? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
3.	Тема 3 «Фронтенд: основы React»	Почему в React нельзя мутировать состояние напрямую? Как работает reconciliation и зачем нужны ключи (key) в списках? В чём разница между компонентами

№ п/п	Название раздела, темы	Перечень рассматриваемых вопросов для самостоятельного изучения
		классами и функциональными компонентами? Как передавать данные от родителя к дочернему компоненту? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
4.	Тема 4 «Фронтенд: хуки и жизненный цикл»	Почему зависимости в useEffect должны быть указаны полностью? Как useCallback помогает избежать лишних перерисовок? В каких случаях стоит создавать кастомный хук? Как правильно очищать подписки и таймеры при размонтировании компонента? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
5.	Тема 5 «Фронтенд: маршрутизация и навигация»	Как передавать параметры в динамический маршрут в React Router? В чём разница между Link и программной навигацией через useNavigate? Как защитить маршрут от неавторизованных пользователей? Как реализовать вложенные маршруты (nested routes)? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
6.	Тема 6. «Бэкенд: основы Node.js и Express»	Как работает событийный цикл (event loop) в Node.js? Зачем нужны middleware в Express и в каком порядке они выполняются? Как обрабатывать CORS на стороне сервера? Как правильно возвращать HTTP-статусы при ошибках и успехе? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
7.	Тема 7 «База данных: MongoDB и Mongoose»	В чём разница между вложением (embedding) и ссылками (referencing) в MongoDB? Как создать схему с валидацией в Mongoose? Почему ObjectId используется вместо числовых ID? Как подключиться к MongoDB Atlas из Express-приложения? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
8.	Тема 8 «Аутентификация и авторизация»	Почему JWT не следует хранить в localStorage? Как устроен refresh-токен и зачем он нужен? Как реализовать middleware, проверяющий роль пользователя? В чём разница между аутентификацией и авторизацией? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
9.	Тема 9 «Взаимодействие	Как обрабатывать ошибки сети при использовании

№ п/п	Название раздела, темы	Перечень рассматриваемых вопросов для самостоятельного изучения
	фронтенда и бэкенда»	fetch? Зачем нужен Content-Type: application/json в заголовках запроса? Как отображать индикатор загрузки во время запроса? Как избежать утечки памяти при обновлении состояния после запроса в размонтированном компоненте? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
10.	Тема 10 «Архитектура fullstack-приложений»	Зачем выделять слой сервисов отдельно от контроллеров? Что такое DTO и как он помогает при работе с API? Какие принципы SOLID применимы к JavaScript/TypeScript? Как организовать структуру проекта для лёгкого масштабирования? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
11.	Тема 11. «Современный React»	В чём преимущество React Query перед ручным управлением состоянием? Как useContext может негативно повлиять на производительность? Когда использовать useReducer вместо useState? Как работает кэширование в React Query и как его настроить? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
12.	Тема 12 «Валидация и обработка ошибок»	Почему валидацию данных нужно проводить и на фронтенде, и на бэкенде? Как Zod помогает синхронизировать типы TypeScript и валидацию? Как унифицировать формат ошибок API для всего приложения? Как правильно отображать ошибки валидации пользователю? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
13.	Тема 13 «Тестирование fullstack-приложений»	В чём разница между unit- и интеграционными тестами? Как мокировать API-вызовы в компонентных тестах? Зачем нужен act() в React Testing Library? Как измерить покрытие кода тестами с помощью Jest? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
14.	Тема 14 «Работа с файлами и медиа»	Как multer проверяет тип и размер загружаемых файлов? Почему нельзя доверять расширению файла при загрузке? Как безопасно отображать изображения из пользовательского ввода? Где лучше хранить загруженные файлы – на сервере или в облаке? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2,

№ п/п	Название раздела, темы	Перечень рассматриваемых вопросов для самостоятельного изучения
		ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
15.	Тема 15 «DevOps и инструменты разработчика»	Зачем использовать Docker, если приложение и так запускается локально? Как устроен Dockerfile для Node.js-приложения? Что проверяет ESLint и как он отличается от Prettier? Как настроить pre-commit hook для форматирования кода? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
16.	Тема 16. «Деплой и облачные среды»	Почему переменные окружения нельзя коммитить в Git? Как Vercel автоматически деплоит приложение из GitHub? Как подключить MongoDB Atlas к бэкенду на Render? Что делать, если после деплоя стили ломаются или API недоступно? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
17.	Тема 17 «Безопасность fullstack-приложений»	Как helmet защищает Express-приложение? Почему NoSQL-инъекции возможны в MongoDB и как их предотвратить? Как настроить CORS, чтобы разрешить только свой фронтенд? Что такое Content Security Policy и как она блокирует XSS? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).
18.	Тема 18 «Документирование и профессиональные стандарты»	Как описать REST API в формате OpenAPI? Что обязательно должно быть в README.md проекта? Как настроить ESLint и Prettier, чтобы они не конфликтовали? Зачем нужен package-lock.json и стоит ли его коммитить? (ПКос-3.1, ПКос-3.2, ПКос-3.3, ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-5.1, ПКос-5.2, ПКос-5.3, ПКос-6.1,ПКос-6.2, ПКос-6.3, ПКос-7.1, ПКос-7.2, ПКос-7.3, ПКос-8.1, ПКос-8.2, ПКос-8.3).

5. Образовательные технологии

Таблица 6

Применение активных и интерактивных образовательных технологий

№ п/п	Тема и форма занятия		Наименование используемых активных и интерактивных образовательных технологий (форм обучения)
1.	Тема 1 «Введение в fullstack-разработку и экосистему JavaScript/TypeScript»	ПЗ	Мозговой штурм
2.	Тема 4 «Фронтенд: хуки и жизненный цикл»	ПЗ	Разбор конкретных ситуаций
3.	Тема 9 «Взаимодействие фронтенда и бэкенда»	ПЗ	Разбор конкретных ситуаций
4.	Тема 12 «Валидация и обработка ошибок»	ПЗ	Мозговой штурм
5.	Тема 13 «Тестирование fullstack-приложений»	ПЗ	Разбор конкретных ситуаций
6.	Тема 17 «Безопасность fullstack-приложений»	ПЗ	Разбор конкретных ситуаций

6. Текущий контроль успеваемости и промежуточная аттестация по итогам освоения дисциплины

6.1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений и навыков и (или) опыта деятельности

Вопросы для подготовки к устным опросам

Тема 1. Введение в fullstack-разработку и экосистему JavaScript/TypeScript

1. Какие три основных fullstack-стека вы знаете?
2. Зачем нужен TypeScript, если есть JavaScript?
3. Какая команда инициализирует npm-проект?

Тема 2. Основы TypeScript

1. Как объявить переменную, которая может быть строкой или числом?
2. Что делает ключевое слово async перед функцией?
3. Как проверить, что объект соответствует интерфейсу?
4. Почему ошибка «Object is possibly 'null'» появляется даже при наличии данных?

Тема 3. Фронтенд: основы React

1. Как создать функциональный компонент в React?
2. Как передать данные из родителя в дочерний компонент?

3. Как обновить состояние компонента при нажатии кнопки?
4. Зачем нужны ключи (key) при рендеринге списка?

Тема 4. Фронтенд: хуки и жизненный цикл

1. Какой хук используется для управления состоянием?
2. Как загрузить данные при первом отображении компонента?
3. Что такое побочный эффект (side effect) в React?
4. Как предотвратить утечку памяти при запросе данных?

Тема 5. Фронтенд: маршрутизация и навигация

1. Какой пакет используется для маршрутизации в React?
2. Как перейти на другую страницу по клику на кнопку?
3. Как получить параметр из URL (например, /user/123)?
4. Как выделить активную ссылку в навигации?

Тема 6. Бэкенд: основы Node.js и Express

1. Какая функция создаёт сервер в Express?
2. Как обработать POST-запрос к маршруту /api/users?
3. Как включить обработку JSON в Express?
4. Какой код HTTP возвращается при успешном создании ресурса?

Тема 7. База данных: MongoDB и Mongoose

1. Как подключиться к MongoDB из Express-приложения?
2. Как создать модель пользователя с полями name и email?
3. Как сохранить новый документ в коллекцию?
4. Как найти все документы в коллекции?

Тема 8. Аутентификация и авторизация

1. Что такое JWT и зачем он нужен?
2. Какой HTTP-заголовок используется для передачи токена?
3. Как проверить, что пользователь вошёл в систему, на фронтенде?
4. Какой маршрут обычно используется для входа (/login или /signup)?

Тема 9. Взаимодействие фронтенда и бэкенда

1. Какой метод fetch используется для получения данных?
2. Как преобразовать ответ в формат JSON?
3. Где хранится токен после входа?
4. Как отобразить ошибку, если сервер вернул 400?

Тема 10. Архитектура fullstack-приложений

1. Как называется слой, отвечающий за бизнес-логику?
2. Зачем выделять DTO отдельно от модели?
3. Какой файл обычно содержит настройки подключения к БД?
4. Как избежать дублирования кода в контроллерах?

Тема 11. Современный React

1. Какой хук заменяет Redux для глобального состояния?
2. Что делает React Query при повторном запросе к тем же данным?
3. Как предотвратить лишние перерисовки компонента?
4. Зачем использовать useMemo для сложных вычислений?

Тема 12. Валидация и обработка ошибок

1. Какой пакет используется для валидации на бэкенде?
2. Где лучше всего проверять email – на фронтенде или бэкенде?
3. Как отобразить ошибку под полем ввода?

4. Что возвращается при ошибке валидации (статус 400)?

Тема 13. Тестирование fullstack-приложений

1. Какой фреймворк используется для unit-тестов в Node.js?
2. Как проверить, что компонент отображает текст?
3. Что такое мок (mock) и зачем он нужен в тестах?
4. Как запустить тесты из командной строки?

Тема 14. Работа с файлами и медиа

1. Какой middleware используется для загрузки файлов в Express?
2. Куда сохраняются файлы по умолчанию при использовании multer?
3. Как ограничить загрузку только изображений?
4. Как отобразить загруженное изображение на фронтенде?

Тема 15. DevOps и инструменты разработчика

1. Какой файл описывает контейнер для Docker?
2. Что делает команда docker build?
3. Какой файл настраивает форматирование кода в Prettier?
4. Зачем нужен package-lock.json?

Тема 16. Деплой и облачные среды

1. На какой платформе обычно деплоят React-приложения?
2. Как передать секретный ключ в облачное приложение?
3. Что такое production-сборка и чем она отличается от dev?
4. Как проверить, что бэкенд доступен после деплоя?

Тема 17. Безопасность fullstack-приложений

1. Какой middleware защищает Express от известных уязвимостей?
2. Почему нельзя хранить JWT в localStorage?
3. Как защитить API от частых запросов одного пользователя?
4. Что проверяет CORS и зачем он нужен?

Тема 18. Документирование и профессиональные стандарты

1. Какой формат используется для документирования API?
2. Что обязательно должно быть в README.md?
3. Как проверить код на соответствие стилю с помощью ESLint?
4. Зачем нужна единая настройка форматирования для всей команды?

Критерии оценки: опрос осуществляет из всего списка студентов. За каждый вопрос, студент может получить максимум 10 баллов. 1 студент, в течении устного опроса отвечает только на 1 вопрос. Таким образом, за каждый опрос студент может получить максимум 10 баллов.

Перечень вопросов, выносимых на промежуточную аттестацию (зачет)

1. Почему TypeScript считается более надёжным, чем JavaScript, при разработке крупных приложений?
2. В чём разница между интерфейсом (interface) и типом (type) в TypeScript? Приведите пример, где только один из них применим.
3. Почему в React нельзя мутировать состояние напрямую (например, state.items.push())? Что может пойти не так?
4. Объясните, как работает useEffect с пустым массивом зависимостей. Когда такой эффект выполняется и очищается?

5. Как React Router определяет, какой компонент отобразить при совпадении URL? Что происходит, если ни один маршрут не совпадает?
6. Зачем в Express используется middleware? Приведите пример трёх middleware и опишите их порядок выполнения.
7. Почему в Express важно обрабатывать ошибки централизованно (через middleware с 4 параметрами)?
8. В чём преимущество MongoDB перед реляционными БД для fullstack-приложений с гибкой структурой данных?
9. Как Mongoose обеспечивает валидацию данных перед сохранением в MongoDB?
10. Почему ObjectId используется как _id по умолчанию, и можно ли его заменить?
11. В чём разница между аутентификацией и авторизацией? Приведите пример, где пользователь прошёл аутентификацию, но не авторизован.
12. Почему JWT не следует хранить в localStorage? Какой более безопасный способ хранения токена в браузере?
13. Как работает механизм refresh-токенов и зачем он нужен, если есть access-токен?
14. Как на фронтенде проверить, что JWT-токен истёк, не отправляя запрос на сервер?
15. Почему при использовании fetch для POST-запроса нужно указывать заголовок Content-Type: application/json?
16. Как избежать утечки памяти, если компонент React размонтируется до завершения асинхронного запроса?
17. Как организовать передачу токена авторизации из фронтенда на бэкенд при каждом запросе?
18. Зачем в Express нужно подключать cors() middleware? Что произойдёт, если его не использовать?
19. Как правильно обрабатывать ошибку «E11000 duplicate key» в MongoDB при регистрации пользователя?
20. Почему в REST API для создания ресурса используется HTTP-метод POST, а не PUT?
21. Как реализовать защиту маршрута на фронтенде, чтобы неавторизованный пользователь не мог попасть на страницу профиля?
22. Какие HTTP-статусы должны возвращаться при: успешном создании (201), ошибке валидации (400), отсутствии авторизации (401), запрете доступа (403)?
23. Почему в React ключи (key) в списках должны быть стабильными и уникальными? Что будет, если использовать индекс массива как ключ?
24. Как работает деструктуризация props в функциональном компоненте? Приведите пример.
25. Как организовать обработку формы на фронтенде так, чтобы данные отправлялись на бэкенд только после валидации?
26. Почему при работе с асинхронными операциями в useEffect нельзя напрямую использовать async в коллбэке?

27. Как настроить Express так, чтобы он автоматически парсил JSON из тела запроса?
28. Как подключить MongoDB к Express-приложению с использованием Mongoose? Напишите минимальный код подключения.
29. Как реализовать logout на фронтенде и бэкенде? Нужно ли что-то удалять на сервере?
30. Какие шаги нужно предпринять, чтобы проект корректно работал локально (фронт + бэк + БД)?

Перечень вопросов, выносимых на промежуточную аттестацию (экзамен)

1. Основные fullstack-стеки: MERN, MEAN, сравнение с Python-стеком.
2. Роль TypeScript в обеспечении надёжности fullstack-приложений.
3. Структура проекта на React и Express: организация папок и файлов.
4. Типизация данных в TypeScript: интерфейсы, типы, union и intersection.
5. Асинхронное программирование в TypeScript: Promise, async/await, обработка ошибок.
6. Компонентная архитектура React: принципы построения и переиспользования компонентов.
7. Состояние и жизненный цикл компонентов в React.
8. Хуки в React: useState, useEffect, useCallback, useMemo, правила их использования.
9. Кастомные хуки: цели создания и примеры применения.
10. Маршрутизация в одностраничных приложениях: React Router, динамические и вложенные маршруты.
11. Архитектура серверных приложений на Node.js: событийный цикл, неблокирующий I/O.
12. Принципы построения RESTful API на Express: ресурсы, HTTP-методы, статусы.
13. Middleware в Express: назначение, порядок выполнения, создание собственного middleware.
14. Нереляционные базы данных: особенности MongoDB, документная модель.
15. ODM Mongoose: схемы, модели, валидация, CRUD-операции.
16. Связи между документами в MongoDB: embedding vs referencing.
17. Аутентификация на основе JWT: структура токена, генерация, верификация.
18. Безопасное хранение и передача JWT в fullstack-приложениях.
19. Реализация refresh-токенов и защита от утечек.
20. Взаимодействие фронтенда и бэкенда: форматы данных, обработка ошибок, CORS.
21. Слоистая архитектура fullstack-приложений: контроллеры, сервисы, модели, DTO.
22. Принципы SOLID в контексте JavaScript/TypeScript.
23. Управление состоянием в современном React: Context API, Redux, React Query.
24. Кэширование данных и фоновая синхронизация с помощью React Query.

25. Валидация входных данных: Zod и Joi на бэкенде, React Hook Form на фронтенде.
26. Централизованная обработка ошибок в fullstack-приложениях.
27. Методы тестирования fullstack-приложений: unit, компонентные, интеграционные тесты.
28. Инструменты тестирования: Jest, React Testing Library, SuperTest, Postman.
29. Покрытие кода тестами: метрики, настройка, интерпретация результатов.
30. Загрузка и обработка файлов в Express: middleware multer, безопасность.
31. Хранение медиафайлов: локальное и облачное, ограничения и лучшие практики.
32. Контейнеризация fullstack-приложений с помощью Docker.
33. Сборка и запуск fullstack-приложения через docker-compose.
34. Инструменты обеспечения качества кода: ESLint, Prettier, Husky, lint-staged.
35. CI/CD-процессы в fullstack-разработке: GitHub Actions, автоматическая сборка и тестирование.
36. Деплой фронтенда: Vercel, Netlify – особенности и настройка.
37. Деплой бэкенда: Render, Railway – подключение к MongoDB Atlas, переменные окружения.
38. Основные угрозы безопасности fullstack-приложений: XSS, NoSQL-инъекции, подделка токенов.
39. Защита Express-приложений: middleware helmet, cors, rate limiting.
40. Безопасность на фронтенде: Content Security Policy, безопасная работа с DOM.
41. Аудит зависимостей и управление уязвимостями в npm-проектах.
42. Стандарты документирования API: OpenAPI/Swagger – структура, генерация, интеграция.
43. Профессиональные стандарты оформления кода: именование, комментирование, структура проекта.
44. Управление версиями зависимостей: package.json, package-lock.json, семантическое версионирование.
45. Полный жизненный цикл fullstack-приложения: от идеи до production и поддержки.

Практическое задание № 1
«Типизация данных и асинхронные операции в TypeScript»

Цель: изучить основы статической типизации и асинхронного программирования в TypeScript, научиться работать с интерфейсами, union-типами и обработкой Promise.

Условие: требуется разработать набор утилит для работы с данными учёта сельхозтехники (тракторы, комбайны), включая валидацию и имитацию загрузки из API.

Требуется:

1. Создать базовую структуру проекта:
   a. Инициализировать npm-проект и установить TypeScript.

b. Настроить tsconfig.json с strict-режимом.

2. Определить типы и интерфейсы:
   a. Создать интерфейс Machine с полями: id: string, type: 'tractor' | 'combine', model: string, lastServiceDate: Date.
   b. Создать тип ServiceRecord с полями: machineId: string, date: Date, description: string.
3. Реализовать асинхронные функции:
   a. Написать функцию fetchMachines(): Promise<Machine[]>, имитирующую загрузку данных (с setTimeout).
   b. Написать функцию validateEmail(email: string): boolean, возвращающую true, если email содержит @.
4. Протестировать работу:
   a. Создать файл main.ts, в котором вызываются функции и выводятся результаты в консоль.
   b. Скомпилировать и запустить проект через npx tsc && node dist/main.js.

Практическое задание № 2
«Создание UI-компонентов для приложения учёта техники»

Цель: освоить основы React-разработки: создание компонентов, управление состоянием, обработка событий и условный рендеринг.

Условие: требуется разработать пользовательский интерфейс для отображения и добавления сельхозтехники в системе учёта.

Требуется:

1. Создать базовую структуру проекта:
   a. Инициализировать проект через Vite с шаблоном React + TypeScript.
   b. Установить зависимости и запустить dev-сервер.
2. Реализовать компоненты:
   a. Создать компонент MachineCard, отображающий тип, модель и дату ТО техники.
   b. Создать компонент MachineList, принимающий массив техники и отображающий карточки.
   c. Создать компонент AddMachineForm с полями: тип (выпадающий список), модель (текст), дата ТО (поле ввода).
3. Управление состоянием и взаимодействие:
   a. В корневом компоненте (App.tsx) создать состояние machines: Machine[].
   b. Реализовать обработчик onAddMachine, добавляющий новую технику в список.
   c. Передать обработчик в AddMachineForm и отобразить список через MachineList.
4. Добавить интерактивность:
   a. При пустом поле «Модель» блокировать кнопку «Добавить».
   b. После добавления очищать форму.
   c. Отображать сообщение «Нет техники» если список пуст.

Комплект разноуровневых заданий

Тема 1. «Введение в fullstack-разработку и экосистему JavaScript/TypeScript»

Задание репродуктивного уровня

Опишите разницу между стеками MERN и MEAN. Какие фреймворки используются на фронтенде и бэкенде в каждом из них?

Задание реконструктивного уровня

Студент создал проект, но при запуске npm install возникает ошибка «engines not satisfied». Объясните, почему это происходит, и предложите способы решения.

Тема 2. «Основы TypeScript»

Задание репродуктивного уровня

В чём разница между interface и type в TypeScript? Приведите пример, где можно использовать только interface.

Задание реконструктивного уровня

В коде используется any для типизации ответа API. Объясните, какие риски это несёт, и предложите способ безопасной типизации с использованием interface.

Тема 3. «Фронтенд: основы React»

Задание репродуктивного уровня

Опишите, как работает механизм ключей (key) в списках компонентов React. Почему нельзя использовать индекс массива как ключ при изменяющемся списке?

Задание реконструктивного уровня

При обновлении списка техники в приложении интерфейс «мерцает» – все компоненты пересоздаются. Объясните причину и предложите решение.

Тема 4. «Фронтенд: хуки и жизненный цикл»

Задание репродуктивного уровня

Какие правила существуют для использования хуков в React? Почему нельзя вызывать хуки внутри условий или циклов?

Задание реконструктивного уровня

После запроса данных в useEffect приложение выдаёт ошибку «Can’t perform a React state update on an unmounted component». Объясните, почему это происходит, и предложите исправление.

Тема 5. «Фронтенд: маршрутизация и навигация»

Задание репродуктивного уровня

Как в React Router передать параметр в маршрут (например, /machine/123)? Как получить этот параметр внутри компонента?

Задание реконструктивного уровня

Пользователь заходит на защищённую страницу /profile без авторизации и видит пустой экран. Как реализовать перенаправление на /login в таком случае?

Тема 6. «Бэкенд: основы Node.js и Express»

Задание репродуктивного уровня

Опишите назначение middleware в Express. В каком порядке выполняются middleware, и почему порядок важен?

Задание реконструктивного уровня

При отправке POST-запроса сервер возвращает ошибку 400, хотя данные корректны. Объясните, какая настройка Express может отсутствовать, и как её добавить.

Тема 7. «База данных: MongoDB и Mongoose»

Задание репродуктивного уровня

В чём разница между embedding и referencing в MongoDB? Приведите пример, когда лучше использовать каждый подход.

Задание реконструктивного уровня

При сохранении документа в MongoDB поле email дублируется у разных пользователей. Как с помощью Mongoose запретить дубликаты и обработать ошибку на бэкенде?

Тема 8. «Аутентификация и авторизация»

Задание репродуктивного уровня

Из каких трёх частей состоит JWT-токен? Какая часть содержит данные пользователя?

Задание реконструктивного уровня

После перезагрузки страницы пользователь теряет авторизацию, хотя токен сохранён в localStorage. Объясните, как реализовать автоматический вход при наличии валидного токена.

Тема 9. «Взаимодействие фронтенда и бэкенда»

Задание репродуктивного уровня

Какой HTTP-статус должен возвращать сервер при успешном создании ресурса? Какой – при ошибке валидации?

Задание реконструктивного уровня

При запросе к API возникает ошибка CORS. Объясните, на какой стороне (фронт/бэк) нужно вносить исправления, и как правильно настроить CORS в Express.

Тема 10. «Архитектура fullstack-приложений»

Задание репродуктивного уровня

Какие слои выделяются в слоистой архитектуре fullstack-приложения? Какова ответственность каждого слоя?

Задание реконструктивного уровня

В контроллере содержится бизнес-логика (например, расчёт урожайности). Объясните, почему это нарушает принцип разделения ответственностей, и перенесите логику в сервис.

Тема 11. «Современный React»

Задание репродуктивного уровня

Как React Query кэширует данные по умолчанию? Как долго сохраняются данные без активного использования?

Задание реконструктивного уровня

При каждом вводе в поисковое поле отправляется запрос на сервер, что вызывает нагрузку. Предложите способ оптимизации с использованием React Query и debounce.

Тема 12. «Валидация и обработка ошибок»

Задание репродуктивного уровня

Как Zod помогает поддерживать синхронизацию между TypeScript-типами и валидацией на бэкенде?

Задание реконструктивного уровня

Ошибки валидации с бэкенда приходят в разном формате, и фронтенд не может их отобразить. Предложите единый формат ответа и способ обработки на фронтенде.

Тема 13. «Тестирование fullstack-приложений»

Задание репродуктивного уровня

В чём разница между unit-тестами и компонентными тестами? Какой инструмент используется для каждого вида в JavaScript-экосистеме?

Задание реконструктивного уровня

Тест на компонент падает, потому что он делает запрос к реальному API. Объясните, как замокать запрос и почему это необходимо.

Тема 14. «Работа с файлами и медиа»

Задание репродуктивного уровня

Как middleware multer определяет, куда сохранять загруженные файлы и какие типы разрешены?

Задание реконструктивного уровня

Пользователь загружает файл с расширением .jpg, но на сервере он исполняется как скрипт. Объясните, почему это произошло, и как проверить MIME-тип файла.

Тема 15. «DevOps и инструменты разработчика»

Задание репродуктивного уровня

Какие два основных файла нужны для контейнеризации fullstack-приложения с помощью Docker?

Задание реконструктивного уровня

При запуске docker-compose up фронтенд не может соединиться с бэкендом. Объясните, как настроить сетевое взаимодействие между контейнерами.

Тема 16. «Деплой и облачные среды»

Задание репродуктивного уровня

Почему переменные окружения не должны храниться в коде, и как их безопасно передавать в облачные сервисы?

Задание реконструктивного уровня

После деплоя на Render бэкенд возвращает 404 на все маршруты. Объясните, какая настройка Express может отсутствовать для работы в production.

Тема 17. «Безопасность fullstack-приложений»

Задание репродуктивного уровня

Как middleware helmet защищает Express-приложение? Назовите три HTTP-заголовка, которые он устанавливает.

Задание реконструктивного уровня

Злоумышленник отправляет запрос с поддельным JWT-токеном и получает доступ к данным. Объясните, как усилить защиту: проверка подписи, срок действия, секретный ключ.

Тема 18. «Документирование и профессиональные стандарты»

Задание репродуктивного уровня

Какие три обязательных раздела должны быть в файле README.md fullstack-проекта?

Задание реконструктивного уровня

Два разработчика коммитят код с разным форматированием, что мешает code review. Предложите решение с использованием ESLint и Prettier.

Критерии оценки: принимаются преподавателем и оцениваются вместе с устными опросами.

6.2. Описание показателей и критериев контроля успеваемости, описание шкал оценивания

Для оценки знаний, умений, навыков и формирования компетенции по дисциплине применяется **балльно-рейтинговая** система контроля и оценки успеваемости студентов.

В основу балльно-рейтинговой системы (БРС) положены принципы, в соответствии с которыми формирование рейтинга студента осуществляется в ходе текущей работы в семестре.

Работы должны быть выполнены по своему варианту, оформлены в соответствии с требованиями стандартов по оформлению текстовых документов в текстовом редакторе MS Word. Работы сдаются в электронном виде.

По результатам защиты могут быть получены следующие баллы:

9-10 баллов – расчеты (если имеются) проведены корректно, результаты правильно интерпретированы. Полностью выполнены все пункты выданного задания. Работа оформлена в соответствии с требованиями стандартов по оформлению текстовых документов. Студент развернуто и свободно ответил на все вопросы при защите работы.

7-8 баллов – работа выполнена, выполнены все пункты выданного задания, но не полностью, либо с несущественными ошибками, имеются незначительные ошибки в интерпретации результатов и/или оформлении. Студент в целом ответил на все поставленные вопросы, ориентируется в работе.

4-6 баллов – работа в целом выполнена, выполнены основные, но не все пункты выданного задания, либо с существенными ошибками, имеются значительные ошибки в интерпретации полученных результатов и представления данных, оформления работы. Некоторые вопросы по работе вызывают затруднения.

1-3 балла – имеются грубые ошибки в методике выполнения, интерпретации полученных результатов и представления данных, оформления работы, большая часть пунктов выданного задания не выполнена. Студент не отвечает на вопросы при защите.

В течение периода обучения по дисциплине студент должен выполнить и защитить 11 практических заданий (индивидуальных или групповых проектов), каждое из которых оценивается максимум на 10 баллов. За посещение занятий добавляется 0,15 балла за каждый час ($68 \cdot 0{,}15$), участие в конференции с до-

кладом по теме, связанной с возможностями практического применения языка Python – 10 баллов. Таким образом, максимально возможная сумма баллов равна: 11 · 10 + 68 · 0,15 + 10 = 110 + 10 + 10 = 130.

Зачет по дисциплине получают студенты, набравшие не менее 60% от максимального количества баллов, т.е. 78 баллов и более.

Итоговая оценка по дисциплине выставляется преподавателем в соответствии со шкалой:

Текущий рейтинг	Оценка			
	«неудовлетворительно»	«удовлетворительно»	«хорошо»	«отлично»
в процентах	0-59	60-69	70-84	85-100
в баллах	0-77	78-90	91-110	111-130

Студенты, набравшие в течение семестра менее 78 баллов, пишут итоговую зачетную работу. К написанию итоговой зачетной работы допускаются студенты, **в случае выполнения всех практических работ**.

7. Учебно-методическое и информационное обеспечение дисциплины

7.1 Основная литература

1. Русляков, А. А. Фронтенд-разработка : учебно-методическое пособие / А. А. Русляков, Н. В. Братусь. – Москва : РТУ МИРЭА, 2025. – 112 с. – ISBN 978-5-7339-2522-6. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/497990 – Режим доступа: для авториз. пользователей.

2. Ермаков, С. Р. Веб-разработка : учебно-методическое пособие / С. Р. Ермаков. – Москва : РТУ МИРЭА, 2025. – 95 с. – ISBN 978-5-7339-2593-6. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/504852 – Режим доступа: для авториз. пользователей.

3. Баланов, А. Н. Бэкенд-разработка веб-приложений: архитектура, проектирование и управление проектами : учебное пособие для вузов / А. Н. Баланов. – 2-е изд., стер. – Санкт-Петербург : Лань, 2025. – 312 с. – ISBN 978-5-507-52472-3. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/451820 – Режим доступа: для авториз. пользователей.

7.2 Дополнительная литература

1. Ермаков, С. Р. Веб-разработка: Практикум : учебное пособие / С. Р. Ермаков. –Москва : РТУ МИРЭА, 2025. – 91 с. – ISBN 978-5-7339-2591-2 – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/504850 – Режим доступа: для авториз. пользователей.

2. Леон, У. Разработка веб-приложения GraphQL с React, Node.js и Neo4j / У. Леон ; перевод с английского А. Н. Киселева. – Москва : ДМК Пресс, 2023. – 262 с. – ISBN 978-5-93700-185-6. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/314975 – Режим доступа: для авториз. пользователей.
3. Розенталс, Н. Изучаем Typescript 3 / Н. Розенталс ; перевод с английского Д. А. Беликова. – Москва : ДМК Пресс, 2019. – 608 с. – ISBN 978-5-97060-757-2. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/131712 – Режим доступа: для авториз. пользователей.
4. Елисеев, А. И. Разработка программных интерфейсов веб-приложений с использованием фреймворка FastAPI : учебное пособие / А. И. Елисеев, Ю. В. Минин. – Тамбов : ТГТУ, 2024. – 81 с. – ISBN 978-5-8265-2821-1. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/472310 – Режим доступа: для авториз. пользователей.
5. Хортон, А. Разработка веб-приложений в ReactJS / А. Хортон, Р. Вайс ; перевод с английского Р. Н. Рагимова. – Москва : ДМК Пресс, 2016. – 254 с. – ISBN 978-5-94074-819-9. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/97339 – Режим доступа: для авториз. пользователей.

8. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. React. – URL: https://react.dev/
2. Современный учебник JavaScript. – URL: https://learn.javascript.ru/
3. TypeScript. – URL: https://www.typescriptlang.org/
4. Mongoose ODM. – URL: https://mongoosejs.com/
5. Цифровые профессии: Искусственный интеллект. – URL: https://steps.2035.university/collections/f6361b9a-ea2e-41b1-a18f-9a2f84a9fcd4
6. Миникурс: основы ООП для JS/TS, React разработчика. – URL: https://stepik.org/course/245362/promo?search=8587862300
7. Фулстек-разработка веб-сервиса на TypeScript, React, Node.js – URL: https://stepik.org/course/210768/promo?search=8587862301
8. Введение в React– URL: https://metanit.com/web/react/1.1.php
9. React – URL: https://ru.legacy.reactjs.org/

9. Перечень программного обеспечения и информационных справочных систем

Таблица 9

Перечень программного обеспечения

№ п/п	Наименование раздела учебной дисциплины	Наименование программы	Тип программы	Автор	Год разработки
1	Тема 1. «Введение в fullstack-разработку и экосистему JavaScript/TypeScript»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
2	Тема 2 «Основы TypeScript»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
3	Тема 3 «Фронтенд: основы React»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
4	Тема 4 «Фронтенд: хуки и жизненный цикл»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
5	Тема 5 «Фронтенд: маршрутизация и навигация»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
6	Тема 6. «Бэкенд: основы Node.js и Express»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
7	Тема 7 «База данных: MongoDB и Mongoose»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
8	Тема 8 «Аутентификация и авторизация»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
9	Тема 9 «Взаимодействие фронтенда и бэкенда»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
10	Тема 10 «Архитектура fullstack-приложений»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012

11	Тема 11. «Современный React»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
12	Тема 12 «Валидация и обработка ошибок»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
13	Тема 13 «Тестирование fullstack-приложений»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
14	Тема 14 «Работа с файлами и медиа»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
15	Тема 15 «DevOps и инструменты разработчика»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
16	Тема 16. «Деплой и облачные среды»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
17	Тема 17 «Безопасность fullstack-приложений»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012
18	Тема 18 «Документирование и профессиональные стандарты»	Excel/ Word/Anaconda/ VSCode	Расчетная/система управления пакетами и дистрибутив	Microsoft/Anaconda Inc.	2007/2012

10. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Таблица 10

Сведения об обеспеченности специализированными аудиториями, кабинетами, лабораториями

Наименование специальных* помещений и помещений для самостоятельной работы (№ учебного корпуса, № аудитории)	**Оснащенность специальных помещений и помещений для самостоятельной работы****
1	2
учебная аудитория для проведения занятий лекционного типа, учебная аудитория для проведения занятий семинарского типа, учебная аудитория для проведения курсового проектирования (выполнения курсовых работ), учебная аудитория для групповых и индивидуальных консультаций, учебная аудито-	1. Компьютер – 29 шт.; 2. Стенд «Сергеев Сергей Степанович 1910-1999» (Инв.№591013/25) – 1 шт.; 3. Огнетушитель порошковый (Инв. №559527) – 1 шт.; 4. Подвесное крепление к огнетушителю (Инв. № 559528) – 1 шт.; 5. Жалюзи (Инв. №1107-221225, Инв. №1107-

рия для текущего контроля и промежуточной аттестации (2й учебный корпус, 102 ауд.)	221225) – 2шт.; 6. Стул – 29 шт.; 7. Стол компьютерный – 28 шт.; 8. Стол для преподавателя – 1 шт.; 9. Доска маркерная (Инв. № 558762/5) – 1 шт.; 10. Трибуна напольная (без инв. №) – 1 шт.
учебная аудитория для проведения занятий семинарского типа, учебная аудитория для проведения курсового проектирования (выполнения курсовых работ), учебная аудитория для групповых и индивидуальных консультаций, учебная аудитория для текущего контроля и промежуточной аттестации, помещение для самостоятельной работы (2й учебный корпус, 106 ауд.)	1. Рабочая станция FORSITE THI516G512G, Российская Федерация A4Tech Fstyler F1512 – 16 шт.; 2. Стол наборный (Инв. №410136000010828) – 1 шт. 3. Стол компьютерный (Инв. № 410136000010813-410136000010827) – 15 шт.; 4. Стул (Инв. № 410136000010829-410136000010853) – 25 шт.; 5. Интерактивная панель (Инв. № 410124000603715) – 1 шт.
учебная аудитория для проведения занятий семинарского типа, учебная аудитория для проведения курсового проектирования (выполнения курсовых работ), учебная аудитория для групповых и индивидуальных консультаций, учебная аудитория для текущего контроля и промежуточной аттестации, помещение для самостоятельной работы (2й учебный корпус, 302 ауд.)	1. Компьютер – 16 шт. 2. Телевизор – 1 шт. 3. Стол для преподавателя –1 шт. 4. Стол компьютерный –16 шт. 5. Стул офисный – 17 шт. 6. Компьютер: PRO-3159209 Intel Core i5-10400 2900МГц, Intel B460, 16Гб DDR4, Intel UHD Graphics 630 (встроенная), SSD 240Гб, 500Вт, Mini-Tower – 1 шт. 7. Кондиционер HAIER HSU -24HPL03/R3 (Инв. № 210134000062198) – 1 шт. 8. Вешалка напольная (Инв.№1107-333144, Инв.№1107-333144) – 2 шт.
учебная аудитория для проведения занятий лекционного типа, учебная аудитория для проведения занятий семинарского типа, учебная аудитория для проведения курсового проектирования (выполнения курсовых работ), учебная аудитория для групповых и индивидуальных консультаций, учебная аудитория для текущего контроля и промежуточной аттестации, помещение для самостоятельной работы (2й учебный корпус, 303 ауд.)	1. Трибуна напольная (Инв.№ 599206) – 1 шт.; 2. Жалюзи (Инв.№591110) – 1 шт.; 3. Доска маркетинговая (Инв.№ 35643/4) – 1 шт.; 4. Стол – 15 шт.; 5. Скамейка – 14 шт.; 6. Стол эрго – 1 шт.; 7. Стул – 16 шт.
Центральная научная библиотека имени Н.И. Железнова	Читальные залы библиотеки
Студенческое общежитие	Комната для самоподготовки

11. Методические рекомендации студентам по освоению дисциплины

Приступая к изучению дисциплины «Фронтенд и бэкенд разработка», студенты должны ознакомиться с учебной программой, учебной, научной и методической литературой, имеющейся в библиотеке РГАУ-МСХА им. К.А. Тимирязева, получить в библиотеке рекомендованные учебники и учебно-методические пособия, завести новую тетрадь для работы с первоисточниками.

В ходе занятий вести конспектирование учебного материала. Обращать внимание на категории, формулировки, раскрывающие содержание тех или иных явлений и процессов, научные выводы и практические рекомендации. Задавать преподавателю уточняющие вопросы с целью уяснения теоретических положений, разрешения спорных ситуаций.

В ходе подготовки к практическим занятиям изучить основную литературу, ознакомиться с дополнительной литературой в соответствии с поставленной задачей. При этом учесть рекомендации преподавателя и требования учебной программы. Необходимо дорабатывать свой конспект, делая в нем соответствующие записи из литературы, рекомендованной преподавателем и предусмотренной учебной программой.

При подготовке к экзамену (в конце семестра) повторять пройденный материал в строгом соответствии с учебной программой. Использовать конспекты и литературу, рекомендованную преподавателем. Обратить особое внимание на темы учебных занятий, пропущенных студентом по разным причинам. При необходимости обратиться за консультацией и методической помощью к преподавателю.

Виды и формы отработки пропущенных занятий

Студент, пропустивший занятия обязан самостоятельно подготовиться к теме устного опроса, которые состоялись на практическом занятии. В рамках часов консультаций студент может ответить на вопросы пропущенного устного опроса, которые были пропущены.

12. Методические рекомендации преподавателям по организации обучения по дисциплине

Курс «Фронтенд и бэкенд разработка» должен давать не абстрактно-формальные, а прикладные знания. Данная цель может быть реализована только при условии соблюдения в учебных планах преемственности учебных дисциплин. Базовые знания для изучения Фронтенд и бэкенд разработка дают такие дисциплины, как программирование на Python, Алгоритмизация и программирование, Веб-разработка. Освоение основных тем данной дисциплины позволит студентам сформировать представление о таком сложном предмете как тестирование программного обеспечения, понять всю ширину науки и получить необходимые знания для последующего профессионального развития в этой области.

Студент может подготовить доклад по теме, представляющей его научный интерес, представить результаты в виде презентации. В случае надлежащего качества, его работа может быть заслушана на научном кружке кафедры или

на студенческой научной конференции. По решению кафедры, студенты, занявшие призовые места на научных студенческих конференциях, могут освобождаться от сдачи экзамена по этой дисциплине.

Преподаватель должен указывать, в какой последовательности следует изучать материал дисциплины, обращать внимание на особенности изучения отдельных тем и разделов, помогать отбирать наиболее важные и необходимые сведения из учебных пособий, а также давать объяснения вопросам программы курса, которые обычно вызывают затруднения. При этом преподавателю необходимо учитывать следующие моменты:

1. Не следует перегружать студентов творческими заданиями.
2. Чередовать творческую работу на занятиях с заданиями во внеаудиторное время.
3. Давать студентам четкий инструктаж по выполнению самостоятельных заданий: цель задания; условия выполнения; объем; сроки; требования к оформлению.
4. Осуществлять текущий учет и контроль за самостоятельной работой.
5. Давать оценку и обобщать уровень усвоения навыков самостоятельной, творческой работы.

Программу разработал:

Демичев В.В., канд. экон. наук, доцент _____
(ФИО, ученая степень, ученое звание) (подпись)

Храмов Д.Э., ассистент _____
(ФИО, ученая степень, ученое звание) (подпись)

РЕЦЕНЗИЯ
на рабочую программу дисциплины Б1.В.16 «Фронтенд и бэкенд разработка» ОПОП ВО по направлению 09.03.02 Информационные системы и технологии, направленность «Фуллстек разработка» (квалификация выпускника – бакалавр)

Вахрушевой Инны Алексеевны, доцентом кафедры высшей математики, кандидатом педагогических наук (далее по тексту рецензент), проведено рецензирование рабочей программы дисциплины «Фронтенд и бэкенд разработка» ОПОП ВО по направлению 09.03.02 Информационные системы и технологии**,** направленность «Фуллстек разработка» (бакалавриат) разработанной в ФГБОУ ВО «Российский государственный аграрный университет – МСХА имени К.А. Тимирязева», на кафедре статистики и кибернетики (разработчики – Демичев Вадим Владимирович, доцент, кандидат экономических наук, Храмов Дмитрий Эдуардович, ассистент кафедры статистики и кибернетики).

Рассмотрев представленные на рецензирование материалы, рецензент пришел к следующим выводам:

1. Предъявленная рабочая программа дисциплины «Фронтенд и бэкенд разработка» (далее по тексту Программа) *соответствует* требованиям ФГОС ВО по направлению 09.03.02 Информационные системы и технологии. Программа *содержит* все основные разделы, *соответствует* требованиям к нормативно-методическим документам.

2. Представленная в Программе ***актуальность*** учебной дисциплины в рамках реализации ОПОП ВО *не подлежит сомнению* – дисциплина относится к дисциплинам части, формируемой участниками образовательных отношений – Б1.В

3. Представленные в Программе ***цели*** дисциплины *соответствуют* требованиям ФГОС ВО направления 09.03.02 Информационные системы и технологии.

4. В соответствии с Программой за дисциплиной «Фронтенд и бэкенд разработка» закреплены 6 ***профессиональные компетенции (ПКос)***. Дисциплина «Фронтенд и бэкенд разработка» и представленная Программа *способна реализовать* их в объявленных требованиях.

5. ***Результаты обучения***, представленные в Программе в категориях знать, уметь, владеть *соответствуют* специфике и содержанию дисциплины и *демонстрируют возможность* получения заявленных результатов.

6. Общая трудоёмкость дисциплины «Фронтенд и бэкенд разработка» составляет 8 зачётные единицы (288 часов).

7. Информация о взаимосвязи изучаемых дисциплин и вопросам исключения дублирования в содержании дисциплин *соответствует* действительности. Дисциплина «Фронтенд и бэкенд разработка» взаимосвязана с другими дисциплинами ОПОП ВО и Учебного плана по направлению *09.03.02 Информационные системы и технологии* и возможность дублирования в содержании отсутствует.

8. Представленная Программа предполагает использование современных образовательных технологий, используемые при реализации различных видов учебной работы. Формы образовательных технологий *соответствуют* специфике дисциплины.

9. Программа дисциплины «Фронтенд и бэкенд разработка» предполагает проведение занятий в интерактивной форме.

10. Виды, содержание и трудоёмкость самостоятельной работы студентов, представленные в Программе, *соответствуют* требованиям к подготовке выпускников, содержащимся во ФГОС ВО направления 09.03.02 Информационные системы и технологии.

11. Представленные и описанные в Программе формы *текущей* оценки знаний (устный опрос), *соответствуют* специфике дисциплины и требованиям к выпускникам.

Форма промежуточного контроля знаний студентов, предусмотренная Программой, осуществляется в форме экзамена в восьмом семестре, что *соответствует* статусу дисциплины, как дисциплины части, формируемой участниками образовательных отношений дисциплин учебного плана – Б1.В ФГОС ВО направления 09.03.02.Информационные системы и технологии**.**

12. Формы оценки знаний, представленные в Программе, *соответствуют* специфике дисциплины и требованиям к выпускникам.

13. Учебно-методическое обеспечение дисциплины представлено: основной литературой – 3 источника (базовые учебники), дополнительной литературой – 5 наименования, Интернет-ресурсы – 9 источника и *соответствует* требованиям ФГОС ВО направления *09.03.02 Информационные системы и технологии***.**

14. Материально-техническое обеспечение дисциплины соответствует специфике дисциплины «Фронтенд и бэкенд разработка» и обеспечивает использование современных образовательных, в том числе интерактивных методов обучения.

15. Методические рекомендации студентам и методические рекомендации преподавателям по организации обучения по дисциплине дают представление о специфике обучения по дисциплине «Фронтенд и бэкенд разработка».

ОБЩИЕ ВЫВОДЫ

На основании проведенного рецензирования можно сделать заключение, что характер, структура и содержание рабочей программы дисциплины «Фронтенд и бэкенд разработка» ОПОП ВО по направлению 09.03.02 Информационные системы и технологии, направленность «Фуллстек разработка» (квалификация выпускника – бакалавр), разработанная Демичевым Вадимом Владимировичем, доцентом, кандидатом экономических наук, Храмовым Дмитрием Эдуардовичем, ассистентом кафедры статистики и кибернетики соответствует требованиям ФГОС ВО, современным требованиям экономики, рынка труда и позволит при её реализации успешно обеспечить формирование заявленных компетенций.

Рецензент: Вахрушева Инна Алексеевна, доцент кафедры высшей математики ФГБОУ ВО «Российский государственный аграрный университет – МСХА имени К.А. Тимирязева», кандидат педагогических наук

(подпись)

«26» августа 2025 г.