

Федеральное государственное образовательное бюджетное учреждение
высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ
РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)**

**Кафедра информационных технологий
Факультет информационных технологий и анализа больших данных**

Документ подписан усиленной неквалифицированной электронной подписью
Организация: Финансовый университет при Правительстве РФ
Утверждено: Проректор по учебной и методической работе Е.А. Каменева
Сертификат: pKTyRp27LObOCXrwXyWdjс8YHuGHw/ZF
Дата: 26.11.2025 г.

А.Ю. Шаталова

Современные технологии веб-программирования

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки:

09.03.03 - Прикладная информатика,

Образовательная программа «Прикладные информационные системы в экономике
и финансах»

Профиль:

«Прикладные информационные системы в экономике и финансах»

Рекомендовано

*Факультет информационных технологий и анализа больших данных
(протокол № 03 от 16.12.2025 г.)*

Одобрено

*Кафедра информационных технологий
(протокол № 12 от 03.12.2025 г.)*

© Москва 2026

СОДЕРЖАНИЕ

1.	Наименование дисциплины	3
2.	Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине	3
3.	Место дисциплины в структуре образовательной программы	4
4.	Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	4
5.	Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий	5
5.1.	Содержание дисциплины	5
5.2.	Учебно-тематический план	7
5.3.	Содержание семинаров	8
6.	Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине	10
6.1.	Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	10
6.2.	Перечень вопросов, заданий, тем для подготовки к текущему контролю	12
7.	Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	15
8.	Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	23
9.	Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	23
10.	Методические указания для обучающихся по освоению дисциплины	24
11.	Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем	26
12.	Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	27

1. Наименование дисциплины

«Современные технологии веб-программирования».

2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине

Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
ПКН-1	Способность применять общенаучные, общетехнические знания, математические методы в сфере ИТ	Демонстрирует знания о современных естественнонаучных концепциях, общетехнических подходах, методах математического анализа и моделирования	знать: основы математического анализа, дискретной математики, теории вероятностей; основы физических процессов в ИТ-оборудовании; принципы математического моделирования уметь: применять математические методы для решения типовых ИТ-задач; строить простые математические модели процессов
		Применяет знания для теоретического и экспериментального исследования в сфере разработки программного обеспечения	знать: методы экспериментальной проверки гипотез; основы статистического анализа данных уметь: проводить эксперименты по оценке производительности алгоритмов; анализировать результаты с использованием статистических методов

3. Место дисциплины в структуре образовательной программы

Дисциплина «Современные технологии веб-программирования» относится к «Общефакультетскому (предпрофильному) циклу».

4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся

Заочная форма обучения

Вид учебной работы по дисциплине	Всего (в з/е и часах)	Семестр 5 (в часах)	Семестр 6 (в часах)
Общая трудоёмкость дисциплины	5/180	72	108
Контактная работа- Аудиторные занятия	16	8	8
Лекции	4	2	2
Семинары, практические занятия	12	6	6
Самостоятельная работа	164	64	100
Вид текущего контроля	Контрольная работа, Контрольная работа	Контрольная работа	Контрольная работа
Вид промежуточной аттестации	Зачет, Экзамен	Зачет	Экзамен

5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий

5.1. Содержание дисциплины

Тема 1. ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ И FRONTEND-РАЗРАБОТКА

Введение в архитектуру веб-приложений и математические модели. Клиент-серверная архитектура. Модель HTTP: запрос-ответ, методы, коды состояния, заголовки. Моделирование взаимодействия как конечного автомата. Языки разметки и структурное моделирование данных (HTML5, Semantic Web). Эволюция HTML. Семантическая верстка и её влияние на SEO и доступность (WCAG). Модель DOM-дерева. Принципы визуального дизайна и инженерные подходы к стилизации (CSS3, препроцессоры). Каскад и наследование. Современные модули CSS: Flexbox, Grid (построение математических сеток). Адаптивный и отзывчивый дизайн. Программирование на клиентской стороне: от алгоритмов к фреймворкам (JavaScript/TypeScript). Основы ES6+: структуры данных (Array, Object, Set, Map), функции, замыкания. Асинхронность: Event Loop, Promise, async/await. Моделирование асинхронных процессов. Введение в TypeScript: статическая типизация как метод повышения надежности. Компонентная архитектура и управление состоянием (React/Vue.js). Философия реактивного программирования. Компоненты: жизненный цикл, состояние (state), свойства (props). Управление состоянием на уровне приложения (Context API, Vuex/Pinia).

Тема 2. BACKEND-РАЗРАБОТКА, ДАННЫЕ И БЕЗОПАСНОСТЬ

Проектирование серверной архитектуры и RESTful API. Принципы REST. Модель «ресурс». Проектирование эндпоинтов. Документирование (OpenAPI/Swagger). Альтернативы: GraphQL, gRPC. Математические модели данных и системы управления базами данных. Реляционная модель данных. Нормальные формы (1NF, 2NF, 3NF) как формальные правила для устранения аномалий. Основы реляционной алгебры. SQL и транзакции (ACID). Нереляционные БД (документные, ключ-значение): модель данных, CAP-теорема. Прикладная криптография и безопасность веб-приложений. Модели угроз (STRIDE). Аутентификация и авторизация: JWT (структура, алгоритмы подписи), OAuth 2.0/OpenID Connect. Защита от OWASP Top-10: инъекции, XSS, CSRF. Математические основы: хеш-функции (SHA-256), симметричное/асимметричное шифрование. Интеграция и построение полного стека. Паттерны интеграции frontend и backend. Сетевое взаимодействие: CORS, WebSockets (модель постоянного соединения).

Тема 3. ИНЖЕНЕРИЯ ПРОИЗВОДИТЕЛЬНОСТИ, DEVOPS И РАЗВЕРТЫВАНИЕ

Алгоритмическая оптимизация веб-приложений. Профилирование производительности (Chrome DevTools, Lighthouse). Оптимизация загрузки: чанкинг, lazy loading, tree shaking. Кэширование: стратегии (Cache-Control, ETag), использование Redis. Задача инвалидации кэша. Инженерные практики и системы контроля версий (Git). Промышленный Git: ветвление (Git Flow, GitHub Flow), семантическое версионирование (SemVer). Code Review, принципы чистого кода (SOLID, DRY, YAGNI). Контейнеризация, оркестрация и облачная инфраструктура. Docker: образы, контейнеры, Dockerfile (многоступенчатая сборка). Docker Compose для локальной разработки. Введение в облачные платформы (IaaS, PaaS). Деплой статики и серверных приложений. Непрерывная интеграция и доставка (CI/CD), мониторинг. Принципы CI/CD. Настройка пайплайнов (GitHub Actions/GitLab CI): линтинг, тестирование, сборка, деплой. Основы мониторинга: метрики (Apdex, error rate), логи, алертинг.

5.2. Учебно-тематический план

№ п/п	Наименование тем(разделов) дисциплины	Трудоемкость в часах				
		Всего	Контактная работа - Аудиторная работа			Самостоя тельная работа
			Общая, в т.ч.:	Лекции	Семинары, практическ ие занятия	
1	ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ И FRONTEND- РАЗРАБОТКА	66	6	2	4	60
2	BACKEND-РАЗРАБОТКА, ДАННЫЕ И БЕЗОПАСНОСТЬ	65	5	1	4	60
3	ИНЖЕНЕРИЯ ПРОИЗВОДИТЕЛЬНОСТИ, DEVOPS И РАЗВЕРТЫВАНИЕ	49	5	1	4	44
	Итого	180	16	4	12	164

5.3. Содержание семинаров

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ И FRONTEND-РАЗРАБОТКА	Архитектура веб-приложений и протокол HTTP. Языки разметки и DOM. Принципы стилизации и CSS. Клиентский JavaScript и TypeScript. Компонентная архитектура и состояние.	Семинар, практическое занятие
BACKEND-РАЗРАБОТКА, ДАННЫЕ И БЕЗОПАСНОСТЬ	Архитектура серверных приложений и API. Каковы ключевые принципы (ограничения) архитектурного стиля REST? Как они соотносятся с моделью HTTP? Что подразумевается под «ресурсом» в REST? Как этот концепт воплощается в проектировании эндпоинтов (URL, методы)? Для чего необходимо документирование API? Каковы возможности и структура спецификации OpenAPI (Swagger)? В каких сценариях использование GraphQL может быть предпочтительнее REST? Каковы его основные концепции (схема, типы, запросы)? Чем подход gRPC отличается от REST и GraphQL? Каковы его преимущества и типичные области применения? Модели данных и системы управления базами данных. Что такое транзакция? Раскройте смысл свойств ACID и их важность для целостности данных. Безопасность веб-приложений. Как протокол OAuth 2.0 обеспечивает делегированный доступ к ресурсам? Какую задачу решает OpenID Connect поверх OAuth 2.0?	Семинар, практическое занятие
ИНЖЕНЕРИЯ ПРОИЗВОДИТЕЛЬНОСТИ, DEVOPS И РАЗВЕРТЫВАНИЕ	Оптимизация производительности веб-приложений: Какие инструменты (Chrome DevTools, Lighthouse) используются для профилирования и аудита производительности? На какие ключевые метрики (LCP, FID, CLS) нужно обращать внимание? Промышленная разработка и система контроля версий. Сравните модели ветвления Git Flow и GitHub Flow. Каковы их основные принципы и в каких сценариях каждая модель предпочтительнее? Что такое семантическое версионирование (SemVer, MAJOR. MINOR. PATCH)? Как по номеру версии понять суть внесённых изменений? Какую роль играет процесс Code Review в разработке? Какие типичные ошибки и антипаттерны стоит в нём выявлять? Как принципы чистого кода (SOLID, DRY, YAGNI) влияют на поддерживаемость и	Семинар, практическое занятие

	надёжность кодовой базы? Приведите краткое описание каждого принципа. Контейнеризация, оркестрация и облачная инфраструктура. Непрерывная интеграция, доставка и мониторинг. Какую информацию несут логи приложения (логи доступа, логи ошибок) и системные логи? Как организовать их централизованный сбор и анализ? Что такое система алертинга и на основе каких правил или пороговых значений она должна срабатывать?	
--	--	--

6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине

6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы

Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ И FRONTEND-РАЗРАБОТКА	Архитектурные паттерны и история. Историческая эволюция веб-технологий: от статических страниц до одностраничных приложений (SPA) и прогрессивных веб-приложений (PWA). Изучение альтернативных протоколов передачи данных поверх TCP/IP (например, WebSocket, WebRTC, Server-Sent Events) и анализ их отличий от HTTP. Использование ARIA (Accessible Rich Internet Applications) атрибутов для улучшения семантики и доступности динамического контента. Особенности верстки для различных типов устройств (ТВ, умные часы, принтеры). Расширенные возможности CSS: Изучение современных и экспериментальных CSS-модулей: контейнерные запросы (@container), каскадные слои (@layer), функции color-mix(), clamp(). Написание сложных анимаций и переходов с использованием @keyframes, свойства animation и рассмотрение их производительности. Принципы и реализация методологии атомарного/функционального CSS (например, Tailwind CSS) и её сравнение с традиционными подходами (БЭМ). Принципы реактивного программирования с использованием библиотеки RxJS (Observable, Observer, операторы). Концепция виртуального DOM (Virtual DOM) и алгоритма согласования (reconciliation). Сравнение с другими подходами (компиляция, как в Svelte).	Работа с литературой, практикум, подготовка к семинару
BACKEND-РАЗРАБОТКА, ДАННЫЕ И БЕЗОПАСНОСТЬ	Глубокий анализ GraphQL: создание и валидация схемы, система типов, механизмы резолверов, N+1 проблема и решения (DataLoader). Сравнение стратегий пагинации (cursor-based vs offset-based). Изучение принципов RPC (Remote Procedure Call) и специфики реализации gRPC: Protocol Buffers (proto-файлы), потоковая передача данных, сравнение производительности с REST/GraphQL. Паттерны проектирования масштабируемых	Работа с литературой, аналитическая работа

	backend-приложений (микросервисы, CQRS, Event Sourcing). Продвинутое концепции баз данных. Углубленная безопасность. Детальный разбор атак и методов защиты: подбор структуры JWT (JWT forgery), подмена ключей (JWK/JWKS), timing-атаки. Изучение протокола OAuth 2.0 на уровне грантов (authorization code, client credentials, device code) и механизмов защиты (state, PKCE). Реализация duplex-связи с использованием Server-Sent Events (SSE) и сравнение с WebSockets. Паттерны обработки ошибок и обеспечения отказоустойчивости в распределенных системах (Retry, Circuit Breaker, Bulkhead). Проектирование API Gateway: маршрутизация, композиция запросов, аутентификация на уровне шлюза.	
ИНЖЕНЕРИЯ ПРОИЗВОДИТЕЛЬНОСТИ, DEVOPS И РАЗВЕРТЫВАНИЕ	Продвинутое оптимизация производительности. Расширенные практики DevOps. Углубленное изучение CI/CD и тестирования. Облачные платформы и масштабирование. Архитектура высокодоступных и отказоустойчивых приложений в облаке: балансировщики нагрузки, автомасштабирование (auto-scaling группы). Работа с облачными сервисами: managed-базы данных, кэши, очереди сообщений, бессерверные вычисления (serverless, AWS Lambda, Cloud Functions). Функции безопасности в облачных средах: IAM (Identity and Access Management), Security Groups, VPC. Анализ и оптимизация стоимости инфраструктуры: использование различных типов инстансов, резервирование ресурсов, бюджеты и алерты. Принципы GitOps: использование Git как единого источника истины для инфраструктуры и конфигураций. Анализ и настройка сборки проекта: ускорение сборки за счет кэширования зависимостей и артефактов. Использование инструментов для анализа покрытия кода (code coverage) и поддержания качества кодовой базы (SonarQube).	Работа с литературой, практикум

6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю

5 семестр

Примерные вопросы контрольной работы

1. Объясните принципы клиент-серверной архитектуры применительно к веб-приложениям. Опишите жизненный цикл HTTP-запроса.
2. В чём заключается философия семантической верстки (HTML5)? Приведите примеры семантических тегов и объясните их влияние на доступность (a11y).
3. Опишите модель каскадирования и наследования в CSS. Как разрешается конфликт стилей при одинаковой специфичности селекторов?
4. Объясните концепцию асинхронности в JavaScript. Что такое Event Loop и как работают Promises/async-await?
5. Раскройте понятия «состояние» (state) и «свойства» (props) в компонентном фреймворке (React/Vue). Как и когда они обновляются?
6. Сформулируйте основные принципы (ограничения) REST. Как они реализуются через HTTP-методы и коды состояния?
7. Что такое SQL-инъекция и XSS-атака? Опишите базовые механизмы защиты от каждой из них.
8. Объясните назначение и различия между сессиями (session) и токенами (JWT) для аутентификации пользователей.

Примерные задания контрольной работы

1. HTML/CSS: По заданному макету (скриншот/figma) сверстайте адаптивную карточку товара с использованием Flexbox/Grid. Карточка должна корректно отображаться на мобильных устройствах.
2. JavaScript: Напишите функцию, которая принимает массив объектов и возвращает новый массив, отсортированный по определённому полю объекта. Реализуйте фильтрацию массива по строковому критерию.

3. JavaScript/DOM: Создайте простой виджет «счётчик» на чистом JS: кнопки «+», «-», отображение текущего значения. Реализуйте сброс значения.
4. React/Vue: Создайте компонент «Список дел» (ToDoList) с возможностью добавления, отметки выполнения и удаления задач. Состояние списка должно храниться внутри компонента.
5. Архитектура/API: Спроектируйте REST API для блога (ресурсы: posts, comments, authors). Опишите эндпоинты (URL + метод) для основных операций (CRUD) над статьями и комментариями к ним.
6. Безопасность: Дан фрагмент кода на Node.js/Python, уязвимый к SQL-инъекции. Перепишите его, используя параметризованные запросы (prepared statements).

6 семестр

Примерные вопросы контрольной работы

1. Каковы ключевые метрики производительности веб-приложения (Core Web Vitals)? Как их измерить и какие есть стратегии для их оптимизации?
2. Опишите процесс и основные этапы CI/CD пайплайна. Какие задачи решает каждый этап (линтер, тесты, сборка, деплой)?
3. В чём суть контейнеризации приложений? Объясните разницу между образом Docker и контейнером. Для чего нужен Docker Compose?
4. Опишите модель ветвления Git Flow. Каковы назначения веток main, develop, feature/*, release/*?
5. Что такое кэширование и зачем оно нужно? Опишите различия между клиентским кэшированием (браузер) и серверным (Redis, Memcached).
6. Объясните принципы горизонтального и вертикального масштабирования приложений. Какие преимущества и ограничения у каждого подхода?
7. Какие основные типы тестирования применяются в веб-разработке (unit, integration, e2e)? Приведите примеры для каждого типа.
8. Что такое «бессерверная» архитектура (Serverless, FaaS)? Назовите её ключевые преимущества и ограничения.

Примерные задания контрольной работы

1. Оптимизация: Проанализируйте предоставленный веб-сайт с помощью Lighthouse в Chrome DevTools. Составьте отчет с выявленными проблемами производительности и предложите конкретные шаги по их исправлению.
2. Docker: Напишите Dockerfile для простого Node.js/Python веб-приложения. Добавьте инструкции для копирования кода, установки зависимостей и запуска приложения.
3. Git: Описан конфликт слияния (merge conflict) в файле. Дайте пошаговую инструкцию, как его безопасно разрешить с помощью команд Git.
4. CI/CD: Опишите конфигурацию (на псевдокоде или YAML для GitHub Actions/GitLab CI) простого пайплайна, который запускает линтер и юнит-тесты при пуше в ветку main.
5. Архитектура: Спроектируйте высокоуровневую архитектуру веб-приложения с прогнозируемой высокой нагрузкой. Укажите компоненты (балансировщик, кэш, несколько серверов приложений, СУБД) и обоснуйте их выбор.
6. Тестирование: Напишите 2-3 юнит-теста (на Jest/Pytest или аналоге) для функции, которая валидирует email-адрес.

Критерии балльной оценки различных форм текущего контроля успеваемости содержатся в соответствующих методических рекомендациях Кафедры информационных технологий Факультета информационных технологий и анализа больших данных.

7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

Перечень компетенций с указанием индикаторов их достижения в процессе освоения образовательной программы содержится в разделе 2. *Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине.*

Типовые контрольные задания или иные материалы, необходимые для оценки индикаторов достижения компетенций, умений и знаний

ПКН-1 Способность применять общенаучные, общинженерные знания, математические методы в сфере ИТ

1) Демонстрирует знания о современных естественнонаучных концепциях, общинженерных подходах, методах математического анализа и моделирования

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: основы математического анализа, дискретной математики, теории вероятностей; основы физических процессов в ИТ-оборудовании; принципы математического моделирования

Уметь: применять математические методы для решения типовых ИТ-задач; строить простые математические модели процессов

Типовые контрольные задания

Построить математическую модель обработки запросов к серверу с использованием теории массового обслуживания.

2) Применяет знания для теоретического и экспериментального исследования в сфере разработки программного обеспечения

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: методы экспериментальной проверки гипотез; основы статистического анализа данных

Уметь: проводить эксперименты по оценке производительности алгоритмов; анализировать результаты с использованием статистических методов

Типовые контрольные задания

Провести сравнительный анализ алгоритмов сортировки по времени выполнения, оформить выводы с применением статистики

Примеры практико-ориентированных заданий

1. Оптимизация SQL-запроса (Backend/Базы данных)

В приложении есть страница профиля пользователя, которая загружается медленно. Анализ показал, что проблема в запросе, который собирает статистику по пользователю.

Исходный "медленный" запрос:

sql

```
SELECT u.name, (SELECT COUNT(*) FROM posts p WHERE p.user_id =  
u.id) as post_count, (SELECT COUNT(*) FROM comments c WHERE  
c.user_id = u.id) as comment_count, (SELECT COUNT(*) FROM likes l  
WHERE l.user_id = u.id) as like_countFROM users uWHERE u.id = 123;
```

Code

Задание:

1. Проанализируйте, почему этот запрос может быть неэффективным.
2. Перепишите его, используя JOIN и агрегатные функции (COUNT, GROUP BY), чтобы получить тот же результат за один проход по таблицам.
3. Какие индексы вы предложите создать для ускорения нового запроса? Укажите поля и типы индексов.

2. Рефакторинг компонента React (Frontend)

Вам передали на доработку компонент, который отображает список товаров. Код имеет проблемы с производительностью и читаемостью.

Исходный код компонента:

javascriptx

```
function ProductList({ products, onSelect }) { const [filter,
setFilter] = useState(''); const filteredProducts =
products.filter(p =>
p.name.toLowerCase().includes(filter.toLowerCase()) ); return
( <div> <input type="text" placeholder="Поиск..." onChange={(e) =>
setFilter(e.target.value)} /> {filteredProducts.map((product,
index) => ( <div key={index} onClick={() => onSelect(product)}>
<h3>{product.name}</h3> <p>Цена: {product.price} руб.</p> <p>В
наличии: {product.stock} шт.</p> <p>Рейтинг:
{'★'.repeat(product.rating)}</p> </div> ))} </div> );}
```

Code

Задание:

1. Найдите и опишите не менее трех проблем в этом компоненте (производительность, потенциальные баги, плохие практики).
2. Предложите и реализуйте исправления (можно описать или показать исправленный код). Учтите: использование стабильных ключей, мемоизацию, оптимизацию рендеринга, обработку edge-cases.
3. Проектирование API эндпоинта (Backend/Архитектура)

Необходимо разработать эндпоинт для создания заказа в интернет-магазине.

Исходные данные:

- У заказа есть статус (pending, paid, shipped, cancelled).
- Заказ может содержать несколько товаров с указанием количества.
- Необходима проверка наличия товара на складе.
- При успешном создании нужно резервировать товар и генерировать номер заказа.

Задание:

1. Спроектируйте структуру HTTP-запроса (метод, URL, тело в формате JSON) и успешного ответа.
2. Опишите пошаговый алгоритм обработки этого запроса на сервере (псевдокод или текстовое описание).
3. Какие HTTP-коды статуса и сообщения об ошибках следует вернуть в случаях: а) товара нет в наличии; б) неверный формат данных; в) успешное создание?

4. Как обеспечить атомарность операции (чтобы не получилось так, что заказ создан, но товар не зарезервировался)?

4. Диагностика и решение проблемы производительности (DevOps/Fullstack)

После развертывания новой версии приложения на production мониторинг показал аномально высокую нагрузку на CPU (90%) и увеличение времени отклика API с 50 мс до 1500 мс.

Данные из мониторинга (Grafana):

- Резко выросло количество запросов к эндпоинту GET /api/v1/products/{id}/recommendations.
- В логах приложения появились ошибки Timeout connecting to Redis.
- База данных показывает нормальную нагрузку.

Задание:

1. Сформулируйте 3 наиболее вероятные гипотезы о причине проблемы.
2. Для каждой гипотезы предложите 1-2 конкретных действия по проверке (какие логи смотреть, какие команды выполнить).
3. Предположим, проблема в "эффekte топа" (случился всплеск трафика на конкретный товар, и кэш рекомендаций для него истек). Какие немедленные (short-term) и долгосрочные (long-term) решения вы можете предложить?

5. Расчет емкости и планирование инфраструктуры (Системное проектирование)

Вы проектируете инфраструктуру для нового сервиса стриминга аудио. Ожидается 1 млн активных пользователей в пик.

Известно:

- Средний размер одного трека: 5 МБ.
- Пиковая нагрузка: 20% пользователей слушают музыку одновременно.
- Пропускная способность одного сервера приложения (Node.js): 1000 RPS.
- Один сервер кэша (Redis) может хранить 10 000 популярных треков.

Задание:

1. Рассчитайте требуемую пиковую пропускную способность сети (в Мбит/с).

2. Оцените минимальное количество серверов приложения.
3. Спроектируйте упрощенную схему развертывания: сколько нужно серверов кэша, как организовать хранение основного архива (в объектном хранилище), куда направить трафик (CDN).
4. Предложите две метрики (помимо загрузки CPU), которые нужно мониторить в первую очередь для этого сервиса.

Примерные вопросы для подготовки к зачету, экзамену

Примерные вопросы для подготовки к зачету

1. Принципы клиент-серверного взаимодействия. Модель HTTP (методы, коды состояния, заголовки).
2. Семантическая вёрстка (HTML5). Основные семантические теги и их значение для доступности и SEO.
3. Базовые концепции CSS: каскад, наследование, специфичность селекторов. Модель визуального форматирования (box model).
4. Основы JavaScript (ES6+): типы данных, функции, область видимости, замыкания.
5. Асинхронность в JavaScript: Event Loop, Callback, Promise, async/await.
6. Компонентный подход во frontend-фреймворках (на примере React или Vue). Понятия компонента, состояния (state) и свойств (props).
7. Основы проектирования RESTful API. Ресурсно-ориентированный дизайн.
8. Реляционная модель данных. Основы SQL (SELECT, INSERT, UPDATE, DELETE, простые JOIN).
9. Основные угрозы безопасности веб-приложений (инъекции, XSS, CSRF) и базовые меры защиты.
10. Принципы работы системы контроля версий Git (commit, branch, merge, remote).

Примерные вопросы для подготовки к экзамену

1. Аутентификация и авторизация в веб-приложениях. Сравните подходы на основе сессий и JWT-токенов.
2. Механизм работы JSON Web Token (JWT). Из каких частей состоит токен и как обеспечивается его целостность?
3. OAuth 2.0: основные роли (клиент, ресурсный сервер, сервер авторизации) и суть потока Authorization Code.
4. Основные угрозы безопасности веб-приложений (OWASP Top 10). Детально опишите механизм XSS-атаки и способы защиты.
5. SQL-инъекция: принцип, пример уязвимого кода и способы защиты (параметризованные запросы, ORM).
6. CSRF-атака (межсайтовая подделка запроса). Объясните механизм и методы защиты (CSRF-токены, SameSite cookies).
7. Система контроля версий Git. Опишите процесс работы с ветками (ветвление, слияние, разрешение конфликтов).
8. Принципы тестирования программного обеспечения. Сравните модульное (unit), интеграционное (integration) и сквозное (e2e) тестирование.
9. Контейнеризация с Docker. Объясните разницу между образом, контейнером и реестром.
10. Назначение и структура Dockerfile и docker-compose.yml. Приведите пример базового Dockerfile для веб-приложения.
11. Непрерывная интеграция и поставка (CI/CD). Какие задачи решает и как выглядит типичный пайплайн?
12. Метрики производительности веб-приложений (Core Web Vitals: LCP, FID, CLS). Что они измеряют и как их улучшить?
13. Кэширование в веб-приложениях. Какие стратегии кэширования вы знаете (на стороне клиента и сервера)?
14. Межсайтовые запросы (CORS). Почему браузер блокирует такие запросы по умолчанию и как настроить сервер для их разрешения?
15. Безопасные заголовки HTTP (HSTS, CSP, X-Frame-Options). Для чего они используются?

16. Процесс деплоя веб-приложения. Опишите этапы от сборки проекта до его развертывания на облачном сервере.

17. Мониторинг приложения после деплоя. Какие ключевые метрики важно отслеживать (uptime, error rate, latency)?

18. Архитектурные решения для обеспечения отказоустойчивости и масштабируемости веб-приложения.

19. Структура и требования к итоговому проекту. Какие разделы должны быть включены в техническую документацию и презентацию?

*Промежуточная аттестация обучающихся проводится в формате тестирования

Аутентификация и авторизация. Сравните механизмы аутентификации на основе

Безопасность и тестирование. Дан фрагмент кода серверного маршрута на Node.js

в) Сформулируйте два модульных теста (unit test) для безопасной версии функции

Архитектура и DevOps. Вы завершили разработку небольшого full-stack приложения

8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины

Основная литература:

1. Шаталова, А.Ю. Основы веб-разработки : Учебное пособие / А.Ю. Шаталова, М.В. Коротеев Электрон. дан. Москва : КноРус, 2025 282 с. Режим доступа: book.ru Internet access <https://book.ru/book/957272> ISBN 978-5-406-14458-9. [БИК ID: RU\bookru\bibl\957272]

Дополнительная литература:

1. Лехмус, М.Ю. Тестовая поддержка базовых технологий веб-программирования. Часть 2 : Учебное пособие / М.Ю. Лехмус, З.Ф. Исхаков Электрон. дан. Москва : Русайнс, 2024 151 с. Режим доступа: book.ru Internet access <https://book.ru/book/955947> ISBN 978-5-466-07897-8. [БИК ID: RU\bookru\bibl\955947]

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

1. <https://www.planetaexcel.ru/>
2. Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/> (<http://library.fa.ru/files/elibfa.pdf>)
3. Электронно-библиотечная система BOOK.RU <http://www.book.ru>

10. Методические указания для обучающихся по освоению дисциплины

1. Общие рекомендации

Дисциплина «Современные технологии веб-программирования» носит прикладной, практико-ориентированный характер. Успешное освоение требует не пассивного запоминания, а активной разработки, экспериментов и решения задач.

- Принцип «Learning by Doing»: Основной метод изучения — практика. Код, который вы не написали сами, считается неизученным. Регулярно выполняйте все практические задания, даже необязательные.
- Системность: Веб-разработка — это «полный стек» (frontend, backend, базы данных, инфраструктура). Старайтесь понимать взаимосвязи между технологиями, а не изучать их изолированно.
- Самостоятельность и инициатива: Технологии меняются стремительно. Используйте лекции как карту, а для углубленного изучения активно привлекайте внешние ресурсы: официальная документация (MDN Web Docs, React Docs), онлайн-курсы (Stepik, freeCodeCamp), профессиональные блоги и форумы (Stack Overflow, Habr).
- Работа в инструментальной среде: С первого дня осваивайте профессиональный инструментарий:
- Система контроля версий Git (обязательно создайте аккаунт на GitHub/GitLab).
- Интегрированная среда разработки (IDE) – Visual Studio Code, WebStorm и др.
- Инструменты разработчика (DevTools) в браузере.

2. Методические указания по видам учебной работы

2.1. Подготовка к лекциям:

- Перед лекцией: Просмотрите тему предыдущей лекции и ознакомьтесь с анонсом новой.
- Во время лекции: Не старайтесь конспектировать «слово в слово». Фиксируйте ключевые концепции, термины, схемы архитектур и ссылки на важные ресурсы (документацию, инструменты), которые дает преподаватель.
- После лекции: В течение 24 часов просмотрите конспект, дополните его, найдите и сохраните ссылки на материалы. Если остались вопросы — сформулируйте их для семинара.

2.2. Подготовка и работа на практических занятиях (лабораторных работах):

- Предварительная подготовка: Изучите теоретический материал, необходимый для выполнения задания. Установите и проверьте требуемое программное обеспечение (Node.js, базы данных, библиотеки) заранее.
- На занятии: Активно работайте над заданием. Используйте время для консультации с преподавателем. Не просто получайте готовые решения, а задавайте вопросы «почему так?».
- Оформление результатов: Каждая работа должна сопровождаться:

1. Кодом в репозитории Git (с понятными коммитами).
 2. Кратким отчетом (README.md в репозитории или отдельный файл), содержащим: цель работы, описание выполненных шагов, скриншоты результата, ответы на контрольные вопросы, анализ возникших проблем.
- После занятия: Доработайте и «приберите» код, создайте итоговый коммит. Проанализируйте ошибки — они ценнейший источник опыта.

2.3. Выполнение самостоятельной работы:

- Планирование: Разбейте крупный проект (например, «Интернет-магазин») на этапы (верстка, фронтенд на React, бэкенд на Node.js, подключение БД, деплой), согласуйте их с преподавателем.
- Регулярность: Выделяйте время на проект каждую неделю, даже если это 2-3 часа. Это предотвратит «аврал» перед сдачей.
- Документирование и контроль версий: Используйте Git осознанно: создавайте ветки для новых функций (feature/auth), делайте атомарные коммиты с понятными сообщениями («feat: add user login form», «fix: correct API response handling»).
- Поиск решений: Столкнувшись с проблемой, действуйте по алгоритму:

1. Анализ текста ошибки.
 2. Поиск в Google/Stack Overflow (на английском языке эффективность выше).
 3. Обращение к одноклассникам (обсуждение в чате).
 4. Консультация с преподавателем (с готовым описанием проблемы, вашими гипотезами и примерами кода).
3. Рекомендации по работе с информационными ресурсами

- Первоисточники (приоритет):
- MDN Web Docs (Mozilla Developer Network) — лучшая и самая надежная документация по HTML, CSS, JavaScript.
- Официальная документация к фреймворкам и библиотекам (React, Vue, Express.js).
- Актуальные учебные ресурсы:
- freeCodeCamp — интерактивные задачи по веб-разработке.
- Stepik, Coursera — курсы от ведущих вузов и компаний.
- Профессиональное сообщество:
- Stack Overflow — для поиска решений конкретных технических проблем.
- Habr, Medium — для изучения трендов, лучших практик и разборов кейсов.
- Визуализация и практика:
- CodePen, JSFiddle — для быстрых экспериментов с фронтенд-кодом.
- YouTube-каналы (например, Traversy Media, The Net Ninja на англ.; Гоша Дударь, Владилен Минин на русск.) — для наглядных видеоуроков.

4. Критерии успешного освоения дисциплины

Студент может считать дисциплину успешно освоенной, если он:

- Способен самостоятельно: спроектировать, разработать, протестировать и развернуть простое полнофункциональное веб-приложение.
- Понимает жизненный цикл запроса и роль каждой технологии в стеке.
- Имеет публичное портфолио из 3-5 завершенных учебных проектов на GitHub с читаемым кодом и описанием.
- Умеет находить, анализировать и применять современную техническую информацию из авторитетных источников.

11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем

Комплект лицензионного программного обеспечения:

1. Реляционная СУБД
2. Kaspersky
3. Пакет офисных программ
4. Комплект свободно распространяемого программного обеспечения

Современные профессиональные базы данных и информационные справочные системы:

1. Электронная энциклопедия: <http://ru.wikipedia.org/wiki/Wiki>
2. Информационно-правовая система «Гарант»
3. Информационно-правовая система «Консультант Плюс»
4. Система комплексного раскрытия информации «СКРИН» - <http://www.skrin.ru/>

Сертифицированные программные и аппаратные средства защиты информации:

1. не предусмотрены

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

1. **Учебная аудитория** для проведения учебных занятий, предусмотренных программой, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенная оборудованием и техническими средствами обучения: мебель аудиторная (столы, стулья, доска аудиторная), набор демонстрационного оборудования (проектор, экран).

для проведения учебных занятий, предусмотренных