

**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

Сертификат: eZo3/825FskdVC2NPn6pAFaT5YIFX0Vv

Утверждено: Проректор по учебной и методической работе Е.А. Каменева

Дата: 09.02.2026 г.

А.Ю. Шаталова

Основы веб-разработки

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки:

01.03.02 - Прикладная математика и информатика,

Образовательная программа «Прикладное машинное обучение»

Профиль:

«Прикладное машинное обучение»

Рекомендовано

*Факультет информационных технологий и анализа больших данных
(протокол № 06 от 17.03.2026 г.)*

Одобрено

*Кафедра информационных технологий
(протокол № 1 от 27.02.2026 г.)*

Содержание

1. Наименование дисциплины	4
2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине	4
3. Место дисциплины в структуре образовательной программы	6
4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	6
5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий . . .	7
5.1. Содержание дисциплины	7
5.2. Учебно-тематический план	11
5.3. Содержание семинаров	12
6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине	17
6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	17
6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю . . .	20
7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	22
8. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины	26
9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	27
10. Методические указания для обучающихся по освоению дисциплины	28
11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем	31

12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	32
--	----

1. Наименование дисциплины

«Основы веб-разработки».

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

Код компетенции	Наименование компетенции	
ПКП-7	Способность создавать прикладные программные компоненты и интегрировать в них методы анализа данных, машинного обучения и искусственного интеллекта	
Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	
Владеет инструментальными средствами разработки прикладных программных продуктов: языками программирования, библиотеками, фреймворками	знать: синтаксис и основные конструкции современных языков программирования (Python, JavaScript/TypeScript, C#, Java и др.), используемых для создания прикладных решений; назначение и возможности фреймворков для разработки пользовательских интерфейсов (например, React, Vue) и серверной части (например, Express.js, FastAPI, Django); принципы работы с системами контроля версий (Git) для совместной разработки. уметь: писать программный код средней сложности для реализации клиентской и серверной логики; использовать современные библиотеки для упрощения разработки (например, ORM вроде Sequelize или SQLAlchemy); проводить отладку и тестирование созданных компонентов.	

Владеет архитектурными принципами и паттернами создания прикладных программных продуктов на различных платформах	знать: основные принципы проектирования архитектуры ПО (многослойная, клиент-серверная, микросервисная); способы организации взаимодействия между компонентами системы (API, обмен сообщениями); особенности разработки для различных платформ (веб, десктоп, мобильные устройства). уметь: проектировать структуру базы данных для хранения информации; разрабатывать и документировать API (REST, GraphQL) для взаимодействия между компонентами; выбирать подходящий стек технологий в зависимости от решаемой задачи.
Демонстрирует понимание побочных процессов, сопровождающих прикладную разработку программного обеспечения в промышленных условиях и владеет соответствующим инструментарием	знать: способы интеграции предобученных моделей машинного обучения в прикладной код (например, загрузка модели в Python-скрипт или использование ONNX); форматы обмена данными (JSON, XML) для передачи результатов анализа; основы побочных процессов разработки ПО: тестирование, документирование, развертывание (деплой). уметь: разрабатывать программные модули, которые собирают данные, передают их в модель машинного обучения и отображают результат пользователю; настраивать процессы непрерывной интеграции и доставки (CI/CD) для автоматизации развертывания интеллектуальных приложений (как показано в РПД "Основы веб-разработки раздел 7); работать с системами мониторинга для сбора обратной связи о работе интегрированных моделей.

3. Место дисциплины в структуре образовательной программы

Дисциплина «Основы веб-разработки» относится к «Модулю ”Прикладное программирование”».

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

Очная форма обучения

Вид учебной работы по дисциплине	Всего (в з/е и часах)	Семестр 6 (в часах)
Общая трудоёмкость дисциплины	3/108	108
Контактная работа – Аудиторные занятия	50	50
Лекции	16	16
Семинары, практические занятия	34	34
Самостоятельная работа	58	58
Вид текущего контроля	Контрольная работа	Контрольная работа
Вид промежуточной аттестации	Зачет	Зачет

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. Основы веб-технологий и клиентская разработка

Введение в веб-разработку

Архитектура веб-приложений: клиент-серверное взаимодействие. Протокол HTTP: структура запроса и ответа, методы, коды состояния, заголовки. URL и URI: структура, кодирование, параметры запроса. Обзор инструментов веб-разработчика: браузерные средства отладки, IDE (WebStorm, VS Code), системы контроля версий (Git). Профессии в веб-разработке: frontend, backend, fullstack, DevOps.

Язык гипертекстовой разметки HTML

Синтаксис HTML: теги, атрибуты, вложенность. Структура HTML-документа: `<!DOCTYPE>`, `<html>`, `<head>`, `<body>`. Семантическая верстка HTML5: `<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<aside>`, `<footer>`. Работа с текстом: заголовки, абзацы, списки, цитаты. Гиперссылки и якоря: абсолютные и относительные пути. Изображения и мультимедиа: `<img>`, `<figure>`, `<figcaption>`, `<audio>`, `<video>`. Таблицы: `<table>`, `<tr>`, `<td>`, `<th>`, объединение ячеек. Формы и элементы ввода: `<form>`, `<input>`, `<textarea>`, `<select>`, `<button>`, валидация на стороне клиента. DOM (Document Object Model): представление HTML-документа в виде дерева объектов.

Каскадные таблицы стилей CSS

Способы подключения стилей: инлайновые, внутренние, внешние. Селекторы: базовые (тег, класс, id), комбинированные, контекстные, псевдоклассы (`:hover`, `:active`, `:nth-child`), псевдоэлементы (`::before`, `::after`). Каскадность, наследование и специфичность. Блочная модель: `margin`, `border`, `padding`, `width`, `height`, `box-sizing`. Позиционирование элементов: `static`, `relative`, `absolute`, `fixed`, `sticky`. Flexbox: основная ось, поперечная ось, свойства контейнера (`display: flex`, `justify-content`, `align-items`, `flex-wrap`) и элементов (`flex-grow`, `flex-shrink`, `order`). CSS Grid: определение сетки, размещение элементов по ячейкам. Адаптивная верстка: медиа-запросы, резиновая верстка, mobile-first подход. CSS-препроцессоры: обзор возможностей Sass/SCSS (переменные, вложенность, миксины). CSS-фреймворки: Bootstrap (сетка, компоненты, утилиты).

Клиентское программирование на JavaScript

Основы синтаксиса: переменные (`let`, `const`), типы данных, операторы. Управляющие конструкции: условные операторы, циклы. Функции: объявление, функциональные

выражения, стрелочные функции, параметры, замыкания. Объекты и массивы: создание, доступ к свойствам, деструктуризация, spread-оператор. Встроенные объекты: Date, Math, JSON. Обработка событий: добавление и удаление обработчиков, объект события, всплытие и перехват. Работа с DOM: поиск элементов (querySelector, getElementById), изменение содержимого и атрибутов, создание и удаление узлов. Асинхронность: колбэки, промисы, async/await. Fetch API: выполнение HTTP-запросов к серверу, обработка ответов. Введение в TypeScript: статическая типизация, интерфейсы, компиляция в JavaScript.

Развертывание статических сайтов

Понятие хостинга: виды хостинга (статический, виртуальный, выделенный, облачный). Деплой на статические хостинги: GitHub Pages, Netlify, Vercel. FTP-клиенты: загрузка файлов на сервер. Основы мониторинга и отладки: инструменты разработчика в браузере, анализ производительности.

Тема 2. Клиентские фреймворки и современный фронтенд

Введение в frontend-фреймворки

Назначение frontend-фреймворков: управление состоянием, компонентный подход, реактивность. Обзор популярных фреймворков: React, Vue, Angular (сравнение, области применения).

Фреймворк React.js

Установка и настройка проекта (Create React App, Vite). JSX: синтаксис, встраивание выражений, условный рендеринг. Компоненты: функциональные и классовые, пропсы (props), композиция компонентов. Состояние (state): хук useState, обновление состояния, подъем состояния. Обработка событий в React. Жизненный цикл компонента: хук useEffect (выполнение побочных эффектов, подписки, очистка). Работа с формами: управляемые и неуправляемые компоненты. Стилизация в React: инлайн-стили, CSS-модули, CSS-in-JS (styled-components). Маршрутизация: React Router (настройка роутера, ссылки, параметры URL). Управление состоянием приложения: Context API, обзор Redux (стейт, экшены, редьюсеры).

Разработка одностраничных приложений (SPA)

Архитектура SPA: преимущества и недостатки, навигация без перезагрузки страницы. Взаимодействие с сервером: получение данных с API, обработка загрузки и ошибок. Оптимизация производительности: ленивая загрузка компонентов, мемоизация.

Тема 3. Серверная разработка и интеграция с ML

Серверное программирование на Node.js

Платформа Node.js: архитектура, событийно-ориентированная модель, модульная система (CommonJS, ES-модули). Менеджер пакетов npm: установка зависимостей, семантическое версионирование, скрипты. Создание простого HTTP-сервера: встроенный модуль http.

Веб-фреймворк Express.js

Установка и настройка Express-приложения. Маршрутизация: обработка GET, POST, PUT, DELETE запросов, параметры маршрута. Middleware: понятие, встроенные middleware, создание собственных. Обработка статических файлов. Работа с формами и данными запроса: body-parser, multer. Шаблонизаторы: обзор, интеграция Handlebars или EJS.

Альтернативные серверные фреймворки (ознакомительно)

FastAPI (Python): асинхронность, автоматическая документация OpenAPI. Сравнение подходов: Node.js (JavaScript/TypeScript) vs Python для создания бэкенда.

Работа с базами данных

Реляционные и нереляционные базы данных: основные отличия, области применения. Подключение к MongoDB через Mongoose (Node.js) или SQLAlchemy (Python). Выполнение CRUD-операций.

Объектно-реляционное отображение (ORM)

Понятие ORM, преимущества использования. ORM Sequelize для Node.js: определение моделей, связи между таблицами, миграции. ORM SQLAlchemy для Python: декларативное определение моделей, выполнение запросов.

Разработка API

RESTful API: принципы проектирования (ресурсы, методы, коды ответа). Создание REST API на Express.js и FastAPI. Документирование API: OpenAPI (Swagger). GraphQL как альтернатива REST: основные концепции, сравнение.

Интеграция моделей машинного обучения

Способы интеграции ML-моделей в веб-приложение: прямой импорт модели в код (если модель на том же языке). Развертывание модели как отдельного микросервиса (ML microservice). Использование специализированных платформ (TensorFlow Serving, ONNX Runtime). Создание API-обертки для модели на FastAPI (Python) с загрузкой

предобученной модели (pickle/joblib). Передача данных в модель и получение предсказаний в формате JSON. Асинхронная обработка запросов к модели.

Системы управления содержимым (CMS)

Назначение CMS, обзор популярных решений (WordPress, Joomla, Drupal). Установка и настройка WordPress: выбор темы, установка плагинов. Создание контентного сайта: страницы, записи, меню, пользователи. Интеграция пользовательского кода и стилей в CMS.

Тема 4. Деплой, DevOps и мониторинг

Контейнеризация приложений

Docker: основные понятия (образ, контейнер, Dockerfile, docker-compose). Создание Dockerfile для Node.js и Python-приложений. Оркестрация контейнеров: обзор Kubernetes (понятия pod, service, deployment).

Непрерывная интеграция и доставка (CI/CD)

Принципы CI/CD: автоматизация сборки, тестирования и развертывания. Инструменты: GitHub Actions, GitLab CI, Jenkins. Настройка простого пайплайна для автоматического деплоя приложения на тестовый сервер.

Мониторинг и логирование

Сбор метрик приложения: время ответа, количество запросов, ошибки. Инструменты мониторинга: Prometheus, Grafana. Централизованное логирование: ELK Stack (Elasticsearch, Logstash, Kibana). Мониторинг ML-моделей в production: отслеживание дрейфа данных и качества предсказаний.

Основы безопасности веб-приложений

Типовые уязвимости: SQL-инъекции, XSS, CSRF. Защита API: аутентификация (JWT, OAuth2), валидация входных данных. Безопасность при развертывании: использование переменных окружения для секретов, HTTPS.

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	Основы веб-технологий и клиентская разработка	26	12	4	8	14
2	Клиентские фреймворки и современный фронтенд	26	12	4	8	14
3	Серверная разработка и интеграция с ML	28	14	4	10	14
4	Деплой, DevOps и мониторинг	28	12	4	8	16
	Итого	108	50	16	34	58

Формы текущего контроля успеваемости согласно учебному плану: **Контрольная работа.**

5.3. Содержание семинаров

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Основы веб-технологий и клиентская разработка	1. Эволюция веб-технологий: от статических страниц к динамическим веб-приложениям и SPA. 2. Модель OSI и стек протоколов TCP/IP: место протокола HTTP/HTTPS в стеке. 3. Структура HTTP-запроса и HTTP-ответа: методы (GET, POST, PUT, DELETE и др.), заголовки, коды статуса (1xx, 2xx, 3xx, 4xx, 5xx). 4. URL: схема, домен, порт, путь, параметры запроса, якорь –назначение и примеры. 5. Клиент-серверная архитектура: роль браузера и веб-сервера, статические и динамические ресурсы. 6. Инструменты разработчика в браузере: вкладки Elements, Console, Network, Sources –их назначение и возможности. 7. Системы контроля версий: Git (основные понятия: репозиторий, коммит, ветка, слияние), платформы GitHub/GitLab. 8. Профессиональная среда разработки: критерии выбора IDE (WebStorm, VS Code), полезные плагины и настройки для веб-разработки.	Практические занятия/семинары. Лабораторные работы.

Клиентские фреймворки и современный фронтенд	1. Назначение frontend-фреймворков: Почему чистом JavaScript (vanilla JS) недостаточно для разработки сложных современных интерфейсов? Какие проблемы решают фреймворки? 2. Сравнительный анализ React, Vue и Angular: • Архитектурные особенности каждого фреймворка • Критерии выбора для разных типов проектов • Производительность и размер бандла • Кривая обучения и доступность документации 3. Компонентный подход: • Что такое компонент и почему это основная единица декомпозиции в современном фронтенде? • Функциональные и классовые компоненты в React: в чём различия? • Композиция компонентов vs наследование 4. JSX: • Синтаксис и преимущества JSX • Чем JSX отличается от HTML? • Встраивание JavaScript-выражений в разметку 5. Управление состоянием (state): • Понятие состояния компонента • Хук useState: принципы работы и особенности • Подъем состояния (lifting state up) для совместного использования данных между компонентами 6. Жизненный цикл компонента: • Фазы жизненного цикла: монтирование, обновление, размонтирование • Классовые методы componentDidMount, componentDidUpdate, componentWillUnmount • Хук useEffect как унифицированная замена методам жизненного цикла 7. Пропсы (props): • Передача данных от родительского компонента дочернему	Практические занятия/семинары. Лабораторные работы.
---	---	--

Серверная разработка и интеграция с ML	1. Архитектура Node.js: • Чем Node.js отличается от классических многопоточных серверов (Apache, Tomcat)? • Как работает событийно-ориентированная модель и неблокирующий ввод-вывод (non-blocking I/O)? • Что такое цикл событий (event loop) и как он обрабатывает асинхронные операции? 2. Модульная система: • CommonJS vs ES-модули: синтаксис, различия, область применения. • Как происходит разрешение зависимостей и подключение модулей через require и import? 3. Менеджер пакетов npm: • Структура файла package.json: назначение полей dependencies, devDependencies, scripts. • Что такое семантическое версионирование (semver) и как оно работает? • Отличия npm install от npm ci, когда какой использовать? 4. Создание HTTP-сервера: • Как устроен встроенный модуль http? • Чем res.end() отличается от res.send() (при использовании фреймворков)? • Какие заголовки ответа необходимо устанавливать при возврате разных типов данных (HTML, JSON, файлы)? 1. Назначение Express.js: • Какие задачи решает веб-фреймворк поверх встроенного модуля http? • Архитектура Express-приложения: инициализация, настройка, маршрутизация. 2. Маршрутизация (routing): • Как обрабатываются разные HTTP-методы (GET, POST, PUT, DELETE) в Express? • Что такое параметры маршрута (req.params) и параметры запроса (req.query)? Примеры использования. • Как организовать группировку	Практические занятия/семинары. Лабораторные работы.
---	--	--

Деплой, DevOps и мониторинг	1. Docker: основные понятия • Что такое контейнеризация и чем контейнеры отличаются от виртуальных машин? • Образ (image) и контейнер (container): в чем разница, как они связаны? • Dockerfile: назначение, основные инструкции (FROM, RUN, COPY, CMD, ENTRYPOINT, EXPOSE). • Многоступенчатая сборка (multi-stage builds): зачем и когда применяется? 2. Управление контейнерами • Основные команды Docker для работы с образами и контейнерами (build, run, ps, stop, rm, exec). • Проброс портов и монтирование томов (volumes) для сохранения данных. • Сети в Docker: bridge, host, none – для чего нужны? 3. Docker Compose • Назначение Docker Compose, структура файла docker-compose.yml. • Описание сервисов, зависимостей, сетей и томов. • Запуск многоконтейнерных приложений (например, веб-сервер + БД). 4. Оркестрация контейнеров • Проблемы управления множеством контейнеров. • Kubernetes: базовые понятия (pod, service, deployment, ingress). • Сравнение Docker Swarm и Kubernetes: когда что выбирать? 1. Принципы CI/CD • Что такое непрерывная интеграция (Continuous Integration)? • Непрерывная доставка (Continuous Delivery) vs непрерывное развертывание (Continuous Deployment) – в чем различие? • Типовой CI/CD пайплайн: этапы (checkout, сборка, тестирование, деплой). 2. Инструменты CI/CD • GitHub Actions: структура workflow,	Практические занятия/семинары. Лабораторные работы.
------------------------------------	--	--

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
Основы веб-технологий и клиентская разработка	1. История развития веб-технологий и стандартов W3C. 2. Детальное изучение протокола HTTP/2 и HTTP/3: отличия, преимущества. 3. Расширенные возможности инструментов разработчика (Chrome DevTools): профилирование, аудит, эмуляция устройств. 4. Альтернативные системы контроля версий: Mercurial, SVN (сравнение с Git). 5. Обзор современных форматов данных: JSON, XML, YAML, Protocol Buffers – области применения. 6. Расширенные возможности HTML5: работа с видео/аудио, canvas, геолокация, web storage. 7. Доступность веб-страниц (accessibility): ARIA-атрибуты, роль семантических тегов для людей с ограниченными возможностями. 8. Микроразметка и её влияние на SEO. 9. Шаблонизация на клиенте: основы Handlebars, Mustache. 10. Глубокое изучение CSS-препроцессоров: Sass (переменные, примеси, функции, наследование). 11. Современные методологии именования классов: BEM, OOCSS, SMACSS. 12. Анимации и трансформации в CSS (keyframes, transitions, transforms). 13. CSS-переменные (custom properties) и их преимущества. 14. PostCSS: плагины, autoprefixer.	Работа с источниками литературы, подготовка к семинарам и практическим занятиям

Клиентские фреймворки и современный фронтенд	1. Языки и среды выполнения (Node.js, Python, Java) 2. Основы серверной разработки. 3. Принципы работы серверных приложений. Архитектура серверных систем 4. Node.js. 5. Python. 6. Podman 7. MongoDB 8. Аутентификация и авторизация в Web (JWT, OAuth 2.0) 9. Основы безопасности: SQL-инъекции, XSS, CSRF 10. Мониторинг безопасности	Работа с источниками литературы, подготовка к семинарам и практическим занятиям
Серверная разработка и интеграция с ML	1. Подробное изучение цикла событий (event loop): фазы, микротаски и макротаски. 2. Работа с потоками (streams) в Node.js: Readable, Writable, Duplex, Transform. 3. Кластеризация и балансировка нагрузки в Node.js (модуль cluster). 4. Управление процессами: PM2 как менеджер процессов для production. 5. Альтернативные среды выполнения JavaScript: Deno, Bun –сравнение с Node.js. 6. Расширенные возможности маршрутизации: регулярные выражения в маршрутах. 7. Разработка middleware для обработки ошибок. 8. Использование шаблонизаторов: детальное изучение EJS, Handlebars, Pug (сравнение, синтаксис, возможности). 9. Сессии и куки в Express: управление состоянием пользователя. 10. Защита приложений: helmet, csurf, express-rate-limit. 11. WebSocket-сервер на базе Express и Socket.IO.	Работа с источниками литературы, подготовка к семинарам и практическим занятиям

Деплой, DevOps и мониторинг	1. Интеграция Front-end и Back-end. Сборка проекта (Webpack, Vite). Принципы работы сборщиков 2. Модульность. Оптимизация. Автоматизация. Управление зависимостями — контроль версий библиотек. 3. Dev-окружение — среда разработки 4. Test-окружение — среда тестирования 5. Prod-окружение — продакшен-среда 6. Vite 7. Pre-bundling — предварительная сборка 8. Redux — централизованное хранилище 9. Lighthouse — аудит производительности 10. Система контроля версий Git и платформы (GitHub, GitLab) 11. Бэкапы	Работа с источниками литературы, подготовка к семинарам и практическим занятиям
------------------------------------	--	--

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

Примерные вопросы контрольной работы

1. Опишите архитектуру клиент-серверного взаимодействия.
2. Перечислите основные методы HTTP и их назначение.
3. Какие коды состояния HTTP вы знаете? Приведите примеры для каждой группы (2xx, 4xx, 5xx).
4. Что такое DOM? Как JavaScript взаимодействует с DOM?
5. Охарактеризуйте блочную модель CSS (box model).
6. Чем отличается абсолютное позиционирование от относительного?
7. Что такое Flexbox? Для решения каких задач он применяется?
8. В чем разница между синхронным и асинхронным JavaScript?
9. Что такое промис (Promise) и для чего он используется?
10. Опишите базовый рабочий процесс Git: clone, add, commit, push, pull.
11. В чем преимущества использования frontend-фреймворков (React, Vue, Angular)?
12. Что такое компонент в React? Какие бывают виды компонентов?
13. Чем отличаются props от state?
14. Для чего используется хук useEffect? Приведите примеры.
15. Что такое JSX? Чем он отличается от HTML?
16. Как организовать навигацию в React-приложении?
17. Что такое Redux? В каких случаях его целесообразно использовать?
18. Объясните разницу между Context API и Redux.
19. Как устроен событийно-ориентированный цикл Node.js (event loop)?
20. Что такое middleware в Express.js? Приведите примеры использования.
21. Какие способы аутентификации в веб-приложениях вы знаете?
22. В чем разница между реляционными и нереляционными базами данных?
23. Что такое ORM? Назовите преимущества и недостатки использования ORM.
24. Принципы проектирования RESTful API.
25. Какие существуют способы интеграции ML-модели в веб-приложение?
26. Что такое микросервисная архитектура? Когда она оправдана?
27. Что такое контейнеризация? Чем Docker отличается от виртуальной машины?
28. Основные инструкции Dockerfile: FROM, RUN, COPY, CMD, ENTRYPOINT.
29. Для чего нужен Docker Compose?
30. Что такое CI/CD? Назовите основные этапы CI/CD пайплайна.

31. Какие метрики работы веб-приложения необходимо отслеживать?
32. Что такое SQL-инъекция и как от нее защититься?
33. Объясните принцип XSS-атаки и методы защиты.
34. Для чего используются JWT-токены? Как они устроены?

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

ПКП-7 Способность создавать прикладные программные компоненты и интегрировать в них методы анализа данных, машинного обучения и искусственного интеллекта

1) Владеет инструментальными средствами разработки прикладных программных продуктов: языками программирования, библиотеками, фреймворками

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

Знать: синтаксис и основные конструкции современных языков программирования (Python, JavaScript/TypeScript, C#, Java и др.), используемых для создания прикладных решений; назначение и возможности фреймворков для разработки пользовательских интерфейсов (например, React, Vue) и серверной части (например, Express.js, FastAPI, Django); принципы работы с системами контроля версий (Git) для совместной разработки.

Уметь: писать программный код средней сложности для реализации клиентской и серверной логики; использовать современные библиотеки для упрощения разработки (например, ORM вроде Sequelize или SQLAlchemy); проводить отладку и тестирование созданных компонентов.

Типовые контрольные задания

Создать REST API для управления коллекцией книг.

- Модель книги (название, автор, год).
- Реализовать CRUD-операции с использованием Express и Mongoose.
- Валидировать входящие данные (например, с Joi).
- Написать простой middleware для логирования запросов.

2) Владеет архитектурными принципами и паттернами создания прикладных программных продуктов на различных платформах

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

Знать: основные принципы проектирования архитектуры ПО (многослойная, клиент-серверная, микросервисная); способы организации взаимодействия между компонентами системы (API, обмен сообщениями); особенности разработки для различных платформ (веб, десктоп, мобильные устройства).

Уметь: проектировать структуру базы данных для хранения информации; разрабатывать и документировать API (REST, GraphQL) для взаимодействия между компонентами; выбирать подходящий стек технологий в зависимости от решаемой задачи.

Типовые контрольные задания

- Дан «толстый» контроллер с бизнес-логикой. Выделить слои (Service, Repository), применить Dependency Injection. Показать получившуюся структуру.
- Предложить архитектурные решения для системы с пиковыми нагрузками (кэширование, шардинг БД, CDN). Обосновать выбор.

3) Демонстрирует понимание побочных процессов, сопровождающих прикладную разработку программного обеспечения в промышленных условиях и владеет соответствующим инструментарием

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

Знать: способы интеграции предобученных моделей машинного обучения в прикладной код (например, загрузка модели в Python-скрипт или использование ONNX); форматы обмена данными (JSON, XML) для передачи результатов анализа; основы побочных процессов разработки ПО: тестирование, документирование, развертывание (деплой).

Уметь: разрабатывать программные модули, которые собирают данные, передают их в модель машинного обучения и отображают результат пользователю; настраивать процессы непрерывной интеграции и доставки (CI/CD) для автоматизации развертывания интеллектуальных приложений (как показано в РПД "Основы веб-разработки раздел 7); работать с системами мониторинга для сбора обратной связи о работе интегрированных моделей.

Типовые контрольные задания

1. Работа с системой отслеживания ошибок (Jira): создать задачу, описать баг, пройти по жизненному циклу.
2. Использовать Docker для контейнеризации приложения и docker-compose для локальной разработки.
3. Проанализировать логи (ELK) для поиска причины ошибки.

Примеры практико-ориентированных заданий

Задание 1. Сформировать ТЗ для сервиса онлайн-бронирования билетов

- Определить целевую аудиторию и сценарии использования.
- Выделить 5–7 ключевых функциональных требований (например, выбор места, оплата, уведомления).
- Описать нефункциональные требования (безопасность, производительность).
- Составить UML-диаграмму use case.

Задание 2. Провести предпроектное обследование для системы учёта библиотечных фондов

- Интервьюировать 2–3 «заказчиков» (например, сотрудников библиотеки).
- Выявить болевые точки текущего процесса.
- Сформулировать 3–5 гипотез о том, как IT-решение может их устранить.
- Подготовить отчёт с диаграммой потоков данных (DFD).

Задание 3. Разработать ER-диаграмму для CRM-системы малого бизнеса

Определить сущности: клиенты, заказы, контакты, задачи.

- Задать атрибуты для каждой сущности (например, для «клиента»: ID, имя, email, статус).
- Описать связи между сущностями (один-ко-многим, многие-ко-многим).
- Реализовать схему в SQL (PostgreSQL/MySQL) —создать таблицы и связи.

Примерные вопросы для подготовки к зачету

1. Объясните архитектуру клиент-сервер применительно к веб-приложениям. В чём разница между SSR и SPA?
2. Каково назначение семантических тегов HTML5? Приведите примеры (<header>, <article>, <nav>).
3. Опишите принцип работы CSS Flexbox или Grid. Для решения каких задач макетирования они предназначены?
4. Что такое компонентный подход во frontend-фреймворках? Назовите ключевые преимущества.
5. В чём разница между let, const и var в JavaScript? Что такое область видимости (scope)?
6. Что такое RESTful API? Опишите основные принципы (ресурсы, HTTP-методы, коды состояния).
7. Дайте определение ORM. Какие проблемы он решает и какие может создать (например, проблема N+1)?
8. Объясните разницу между реляционными (PostgreSQL) и нереляционными (MongoDB) базами данных. В каких сценариях предпочтительнее каждая из них?
9. Как работает механизм аутентификации с использованием JWT (JSON Web Tokens)? Опишите процесс от логина до доступа к защищенному ресурсу.
10. Назовите три основные угрозы веб-безопасности (из OWASP Top-10) и базовые меры защиты от каждой.
11. Для чего нужна система контроля версий Git? Объясните смысл команд git commit, git push, git pull.
12. Какую проблему решает контейнеризация (Docker)? Что такое образ (image) и контейнер (container)?

8. Перечень основной и дополнительной 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]
2. Ермаков, С. Р. Основы веб-разработки [Электронный ресурс] : учебное пособие / Ермаков С. Р.,Беляев П. В.,Симонова А. В. Москва : РТУ МИРЭА, 2024 181 с. Книга из коллекции РТУ МИРЭА - Информатика <https://e.lanbook.com/book/420965> ISBN 978-5-7339-2147-1. [БИК ID: RU-LAN-BOOK-420965]

Дополнительная литература:

1. Полуэктова, Наталия Робертовна Разработка веб-приложений : учебник для вузов / Н. Р. Полуэктова. 2-е изд. Электрон. дан. Москва : Юрайт, 2025 204 с (Высшее образование) URL: <https://urait.ru/bcode/567610> (дата обращения: 01.09.2025). Режим доступа: Электронно-библиотечная система Юрайт, для авториз. пользователей <https://urait.ru/bcode/567610> ISBN 978-5-534-18645-1 : 889.00. [БИК ID: RU2fURAIT2f567610]

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

1. Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/>
(<http://library.fa.ru/files/elibfa.pdf>)
2. Электронно-библиотечная система BOOK. RU <http://www.book.ru>
3. • Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/>
4. • Научная электронная библиотека eLibrary.ru <http://elibrary.ru>

10. Методические указания для обучающихся по освоению дисциплины

1. Общие рекомендации

Дисциплина 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 одногруппникам (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.

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. Figma
2. Tilda
3. Реляционная СУБД
4. Редактор кода VS CODE
5. Web сервер node.js версии не ниже 18 LTS
6. Kaspersky

Современные профессиональные базы данных и информационные справочные системы:

1. Информационно-правовая система «Гарант»
2. Информационно-правовая система «Консультант Плюс»
3. Электронная энциклопедия: <http://ru.wikipedia.org/wiki/Wiki>
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. **Компьютерный класс** для проведения учебных занятий, предусмотренных программой, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенный оборудованием и техническими средствами обучения: мебель аудиторная (столы, стулья, доска аудиторная), персональные компьютеры, набор демонстрационного оборудования (проектор, экран)