

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: fzgHDAk/ggBtEx1VuTZkEHfmj2UZ4K7s
Дата: 08.10.2025 г.

А.Ю. Шаталова

Разработка информационных систем с применением веб-технологий

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки:

09.03.04 - Программная инженерия,

Образовательная программа «Разработка и внедрение информационно -
аналитических систем»

Профиль:

«Разработка и внедрение информационно - аналитических систем»

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	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	13
6.1.	Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	13
6.2.	Перечень вопросов, заданий, тем для подготовки к текущему контролю	14
7.	Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	15
8.	Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	21
9.	Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	21
10.	Методические указания для обучающихся по освоению дисциплины	23
11.	Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем	24
12.	Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	24

1. Наименование дисциплины

«Разработка информационных систем с применением веб-технологий».

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
ПКП-1	Способность выполнять разработку информационных систем с применением современных инструментальных средств	Демонстрирует знания основных понятий информационных систем	знать: принципы работы современных инструментальных средств, методологии разработки (Agile/Scrum, Kanban), архитектурные паттерны веб-приложений уметь: анализировать предметную область, формулировать функциональные и нефункциональные требования, выбирать и обосновывать стек технологий для проекта, применять инструменты для проектирования (например, Figma, Draw.io, UML)
		Понимает достоинства и недостатки различных современных инструментальных средств для разработки ИС, может критически анализировать существующие решения	знать: особенности, ограничения и взаимосвязи компонентов выбранного стека технологий (например, Node.js + PostgreSQL + React) уметь: устанавливать, настраивать и интегрировать инструментальные средства и среды выполнения; устранять базовые конфликты зависимостей
		Описывает процесс разработки информационных систем	знать: синтаксис и идиомы используемых языков программирования, принципы ООП/ФП, шаблоны проектирования уметь: писать чистый, поддерживаемый код; использовать возможности IDE и инструментов разработки (отладчик, линтер); осуществлять интеграцию компонентов между собой

		Выполняет программную реализацию информационных систем с учетом требований к применяемым технологиям	знать: принципы и методы деплоя (CI/CD, ручное развертывание), основы администрирования веб-серверов (Nginx, Apache) и сред выполнения (Node.js, Python интерпретатор и т.д.), базовые практики обеспечения отказоустойчивости и мониторинга уметь: настраивать и использовать инструменты для деплоя и оркестрации (Docker, Docker Compose, системы управления конфигурациями), развертывать все компоненты системы (backend, frontend, базу данных) и обеспечивать их корректное взаимодействие в новой среде, выявлять и устранять типовые проблемы, возникающие после развертывания (логирование, анализ ошибок)
--	--	---	--

3. Место дисциплины в структуре образовательной программы

Дисциплина «Разработка информационных систем с применением веб-технологий» относится к «Циклу профиля».

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 5 (в часах)	Семестр 6 (в часах)
Общая трудоёмкость дисциплины	8/288	126	162
Контактная работа- Аудиторные занятия	118	68	50
Лекции	50	34	16
Семинары, практические занятия	68	34	34
Самостоятельная работа	170	58	112
Вид текущего контроля	Проектная работа, Проектная работа	Проектная работа	Проектная 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. Введение и основы Front-end разработки

Архитектура клиент-сервер: Эволюция от SSR к SPA и PWA. RESTful API, GraphQL (обзорно). Протоколы HTTP/1.1, HTTP/2, HTTPS: Жизненный цикл запроса, заголовки, безопасность (CORS, CSRF — основы). Инструментарий разработчика: браузерные DevTools (Elements, Network, Console, Performance). Рабочее окружение: Node.js, npm/yarn/pnpm. Сборщики и менеджеры зависимостей (обзорная роль Vite/Webpack). Система контроля версий Git: базовые команды (init, add, commit, push), работа с GitHub/GitLab. Введение в Agile/Scrum: спринты, пользовательские истории, трекер задач (Trello, Jira). HTML5: Углубленно о семантике (<article>, <section>, <header>, <nav>, <main>). Микроразметка (Schema.org). Формы: валидация на стороне клиента (атрибуты required, pattern, type), доступность форм (ARIA-атрибуты aria-label, aria-describedby). Основы веб-доступности (Web Accessibility, a11y): критерии WCAG, семантическая разметка как основа доступности, навигация с клавиатуры. Верстка семантического каркаса для сквозного проекта. CSS3. Flexbox & Grid: создание сложных, адаптивных макетов. Когда что использовать. Адаптивный дизайн: стратегии (mobile-first, desktop-first), медиа-запросы, относительные единицы (vw, vh, rem, em). Позиционирование, псевдоклассы и псевдоэлементы. Препроцессоры (SASS/SCSS): переменные, вложенность, миксины, модульность (@use, @forward). Методологии именования (БЭМ — теория и практика). Инструменты: автопрефиксер, CSS-модули (обзорно). Стилизация каркаса проекта с использованием SCSS и БЭМ, обеспечение адаптивности. Синтаксис ES6+: let/const, стрелочные функции, деструктуризация, операторы spread/rest, модули (import/export). Работа с DOM: Навигация, манипуляция элементами, эффективные методы (querySelector, createElement). События: Делегирование событий, объект события, предотвращение стандартного поведения. Асинхронность. Promise, цепочки вызовов, обработка ошибок. async/await для написания линейного асинхронного кода. Работа с API: fetch() для отправки GET/POST запросов, обработка ответов (JSON), базовая обработка ошибок сети. Хранение данных на клиенте: localStorage, sessionStorage. Добавление интерактивности в проект: динамическое обновление контента, отправка данных формы (моковый API), сохранение настроек. Почему фреймворки? Проблемы Vanilla JS при масштабировании. Философия React: Декларативный подход, Virtual DOM. Функциональные компоненты и JSX-синтаксис. Props: передача данных и колбэков

между компонентами, проверка типов с PropTypes. State (состояние): хук useState, управление внутренним состоянием компонента. Жизненный цикл и side effects: хук useEffect для работы с API, подписками, таймерами. Основы маршрутизации (React Router): концепция SPA, компоненты BrowserRouter, Routes, Route, Link. Управление состоянием на уровне приложения: контекст React (createContext, useContext) как введение в глобальное состояние. Рефакторинг сквозного проекта на React. Разбиение интерфейса на компоненты (Header, Sidebar, Feed, Widget), настройка маршрутизации для разделов, загрузка данных через fetch в useEffect.

Тема 2. Back-end разработка и базы данных

Модели выполнения: сравнение потоковой (Java) и событийно-ориентированной (Node.js) архитектур. Синхронный vs асинхронный I/O. Критерии выбора языка и фреймворка: производительность, экосистема, поддерживаемость, требования проекта. Архитектурные стили: MVC, Layered Architecture, Clean Architecture (обзорно). Введение в шаблоны проектирования (Design Patterns), релевантные для back-end: Singleton, Factory, Repository, Middleware. Создание минимального сервера на выбранном стеке (Node/Express, Python/FastAPI, Java/Spring Boot). Обоснование выбранного стека для сквозного проекта. Инициализация проекта, настройка окружения, создание первого эндпоинта GET /health. Принципы REST: ресурсы, HTTP-методы, коды состояния, идемпотентность, безопасность методов. Проектирование API. Структура проекта: организация кода по слоям (роутеры, контроллеры, сервисы, модели/репозитории). Валидация входящих данных: использование библиотек валидации (Joi, Zod, class-validator). Обработка ошибок: централизованный error handler, кастомные классы ошибок, возврат структурированных ответов. Логирование (Logging): использование Winston/Pino (для Node.js) или аналогов. Уровни логирования. Реализация CRUD-операций для основной сущности проекта (например, habits) с полной валидацией, пагинацией и документацией в Swagger UI. Сравнение СУБД: реляционные (PostgreSQL) vs Документные (MongoDB) vs Ключ-Значение (Redis). CAP-теорема (обзорно). Реляционные БД и PostgreSQL. ORM (Sequelize, TypeORM, Prisma, Hibernate). Документные БД и MongoDB: документная модель, агрегационные пайплайны. Проектирование схемы БД для проекта. Реализация моделей с помощью ORM/ODM. Написание репозиторийного слоя. Настройка миграций для инициализации БД. Аутентификация vs Авторизация. Сессии (Stateful) vs Токены (Stateless). JWT (JSON Web Tokens): структура (header, payload, signature), процесс выдачи и верификации. OAuth 2.0 / OpenID Connect (OIDC): роли (Resource Owner, Client, Authorization Server), поток Authorization Code (для веб-приложений). Практика интеграции с социальными провайдерами (VK ID, Yandex ID). Ролевая модель доступа (RBAC): реализация middleware для проверки прав (isAuthenticated, isAdmin). Реализация регистрации, логина, выхода. Защита CRUD-маршрутов с помощью JWT. Создание

административной зоны с проверкой ролей. Безопасность (Security). OWASP Top-10: Защита от SQL-инъекций (ORM/параметризованные запросы), XSS (санитизация данных), CSRF (CORS, токены). Защита паролей: Хеширование с "солью" (bcrypt, scrypt). Безопасные заголовки HTTP (Helmet.js для Node.js). Валидация и санитизация всех входящих данных. Производительность (Performance). Балансировка нагрузки (Load Balancing), горизонтальное масштабирование (концепт). Развертывание и DevOps. Контейнеризация приложения с помощью Docker (Dockerfile для back-end и БД). Оркестрация многоконтейнерного приложения с docker-compose.yml (App + DB + Redis). Основы CI/CD: написание пайплайна для автоматического тестирования и деплоя (на примере GitHub Actions/GitLab CI). Работа с переменными окружения (.env файлы) для конфигурации. Добавление безопасности в проект. Созидание Docker-образа приложения. Настройка docker-compose для локального запуска полного стека (API + PostgreSQL + Redis). Написание простого GitHub Actions workflow для прогона линтера и тестов.

Тема 3. Интеграция, инструменты и развертывание

Архитектура связанного приложения. Настройка взаимодействия: CORS на бэкенде, проксирование запросов на этапе разработки. Управление конфигурацией: разные настройки для development, production (переменные окружения для API_URL). Сборка фронтенда для production. Инструменты: углубленное сравнение Vite и Webpack. Почему Vite стал стандартом для новых проектов. Оптимизация сборки: минификация, chunk splitting (разделение кода), tree shaking, сжатие assets (изображения, шрифты). Анализ бандла: использование rollup-plugin-visualizer, webpack-bundle-analyzer для поиска и устранения "тяжелых" зависимостей. Рендеринг на стороне сервера (SSR) и статическая генерация (SSG): Концептуальное введение (Next.js, Nuxt) как следующий шаг после SPA. Настройка Vite в frontend-проекте для работы с переменными окружения. Создание production-сборки. Подключение работающего фронтенда к локальному бэкенду, устранение проблем с CORS. Продвинутый Git. Ветвление: стратегии Git Flow, GitHub Flow. Создание и мерж feature-веток. Интерактивное перебазирование (git rebase -i) для "чистой" истории коммитов. Разрешение конфликтов слияния. Работа с тегами, stash. Платформы (GitHub/GitLab) как центр DevOps. Code Reviews: процесс создания Pull Request (PR) / Merge Request (MR), ревью кода, approval. Issue Tracker и Project Boards: управление задачами в рамках методологии Agile. Защита веток (branch protection rules): требования к прохождению CI, обязательные ревью. Семантическое версионирование (SemVer): Правила обозначения версий MAJOR.MINOR.PATCH. Организация командного репозитория для fullstack-проекта. Создание ветки develop, реализация новой функциональности в отдельной feature-ветке, создание PR с описанием изменений, прохождение peer code review. Философия Docker: "Build once, run anywhere". Образы и контейнеры. Создание эффективных

Dockerfile. Многоступенчатые сборки (multi-stage builds) для уменьшения размера итогового образа. Оптимизация кэширования слоев. Лучшие практики: использование non-root пользователя, минимизация количества слоев. Docker Compose для локальной разработки и продакшена. Оркестрация полного стека: фронтенд (Nginx), бэкенд, БД (PostgreSQL), кэш (Redis). Настройка сетей, volumes для сохранения данных. Переменные окружения в docker-compose.yml. Введение в оркестрацию кластеров (Kubernetes): Базовые понятия (Pod, Service, Deployment) и зачем это нужно. Написание Dockerfile для фронтенда (на базе Nginx) и бэкенда. Создание финального docker-compose.prod.yml, который одной командой поднимает всю полноценную систему. Непрерывная интеграция и доставка (CI/CD): Полный цикл от коммита до продакшена. Написание пайплайнов (GitHub Actions / GitLab CI). Структура YAML-файла, jobs, steps. Кэширование зависимостей для ускорения сборок. Запуск линтеров, unit- и интеграционных тестов. Сборка Docker-образов и их публикация в реестр (Docker Hub, GitHub Container Registry). Облачные платформы для деплоя. PaaS (Platform as a Service): Деплой с минимальной настройкой (Vercel для фронтенда, Railway, Heroku для fullstack). Привязка к Git (автодеплой). IaaS (Infrastructure as a Service): Аренда виртуальной машины (Yandex Cloud Compute, VK Cloud Servers, AWS EC2). Настройка сервера: SSH, Docker & Docker Compose, защита (firewall). Деплой через пайплайн: копирование docker-compose.prod.yml на сервер и запуск. Настройка обратного прокси (Nginx) для serving фронтенда и проксирования API-запросов. Домен и SSL: получение бесплатного сертификата Let's Encrypt (Certbot). Мониторинг и логи (введение): Просмотр логов контейнеров (docker logs). Использование облачных мониторингов (базовые метрики).

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	Введение и основы Front-end разработки	110	52	20	32	58
2	Back-end разработка и базы данных	110	48	22	26	62
3	Интеграция, инструменты и развертывание	68	18	8	10	50
	Итого	288	118	50	68	170

5.3. Содержание семинаров

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Введение и основы Front-end разработки	1. Введение в Web-технологии. Архитектура клиент-сервер. HTTP, HTTPS, методы запросов, коды состояния 2. Базовая разметка: HTML5 (семантические теги, формы) 3. Стилизация: CSS3 (Flexbox, Grid, адаптивный дизайн, препроцессоры SASS/SCSS) 4. Основы JavaScript (ES6+): синтаксис, DOM, события, асинхронность (Promise, async/await), Fetch API 5. Введение во Front-end фреймворки (React/Vue.js): компонентный подход, состояние (state), свойства (props)	Практические занятия/Семинары.Лабораторные работы.
Back-end разработка и базы данных	1. Введение в серверную разработку. Языки и среды выполнения 2. Создание RESTful API 3. Работа с базами данных 4. Аутентификация и авторизация 5. Основы безопасности веб-приложений	Практические занятия/Семинары.Лабораторные работы.
Интеграция, инструменты и развертывание	1. Интеграция Front-end и Back-end 2. Система контроля версий Git 3. Введение в контейнеризацию (Docker) 4. Процесс развертывания (деплой) 5. Введение в CI/CD (Непрерывная интеграция и непрерывное развертывание)	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
Введение и основы Front-end разработки	Введение в Web-технологии. Архитектура клиент-сервер. HTTP, HTTPS, методы запросов, коды состояния. История развития веб-технологийПринципы работы интернетаМодель OSI в веб-разработкеКлиент-серверная архитектураТенденции развитияES6+ особенности. DOM и событияWebpack. Babel. ESLint. Testing Libraries	Исследовательская работа. Аналитический обзор.
Back-end разработка и базы данных	Языки и среды выполнения (Node. js, Python, Java)Основы серверной разработки. Принципы работы серверных приложений. Архитектура серверных системNode. js. Python. PodmanMongoDBАутентификация и авторизация в Web (JWT, OAuth 2. 0)Основы безопасности: SQL-инъекции, XSS, CSRFМониторинг безопасности	Работа с учебной литературой и ресурсами
Интеграция, инструменты и развертывание	Интеграция Front-end и Back-end. Сборка проекта (Webpack, Vite). Принципы работы сборщиковМодульность. Оптимизация. Автоматизация. Управление зависимостями — контроль версий библиотек. Dev-окружение — среда разработкиTest-окружение — среда тестированияProd-окружение — продакшен-средаVitePre-bundling — предварительная сборкаRedux — централизованное хранилищеLighthouse — аудит производительностиСистема контроля версий Git и платформы (GitHub, GitLab)Бэкапы	Исследовательская работа. Аналитический обзор.

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

Примерная тематика проектных работ

1. Система учёта и анализа учебных достижений студентов
2. Платформа для онлайн-курсов с тестированием
3. Сервис для управления проектами (аналог Trello/Asana)
4. Интернет-магазин с персонализированными рекомендациями
5. Система бронирования ресурсов (аудиторий, оборудования)
6. Чат-бот для техподдержки с NLP-элементами
7. Геоинформационная система для анализа городской инфраструктуры
8. Система мониторинга ИТ-инфраструктуры
9. Платформа для краудсорсинга идей
10. Личный финансовый трекер с аналитикой

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-1 Способность выполнять разработку информационных систем с применением современных инструментальных средств

1) Демонстрирует знания основных понятий информационных систем

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

Знать: принципы работы современных инструментальных средств, методологии разработки (Agile/Scrum, Kanban), архитектурные паттерны веб-приложений

Уметь: анализировать предметную область, формулировать функциональные и нефункциональные требования, выбирать и обосновывать стек технологий для проекта, применять инструменты для проектирования (например, Figma, Draw.io, UML)

Типовые контрольные задания

Разработать User Stories для веб-приложения.

Составить API-документации для RESTful сервиса.

Создать технического задания с описанием требований.

2) Понимает достоинства и недостатки различных современных инструментальных средств для разработки ИС, может критически анализировать существующие решения

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

Знать: особенности, ограничения и взаимосвязи компонентов выбранного стека технологий (например, Node.js + PostgreSQL + React)

Уметь: устанавливать, настраивать и интегрировать инструментальные средства и среды выполнения; устранять базовые конфликты зависимостей

Типовые контрольные задания

Выбрать технологический стек для решения конкретной задачи.

Настроить окружение разработки с использованием Docker.

Внедрить системы контроля версий в проект.

3) Описывает процесс разработки информационных систем

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

Знать: синтаксис и идиомы используемых языков программирования, принципы ООП/ФП, шаблоны проектирования

Уметь: писать чистый, поддерживаемый код; использовать возможности IDE и инструментов разработки (отладчик, линтер); осуществлять интеграцию компонентов между собой

Типовые контрольные задания

Разработать фронтенд-компонент с использованием современного фреймворка.

Создать REST API для работы с данными.

Реализовать серверную логику обработки запросов.

4) Выполняет программную реализацию информационных систем с учетом требований к применяемым технологиям

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

Знать: принципы и методы деплоя (CI/CD, ручное развертывание), основы администрирования веб-серверов (Nginx, Apache) и сред выполнения (Node.js, Python интерпретатор и т.д.), базовые практики обеспечения отказоустойчивости и мониторинга

Уметь: настраивать и использовать инструменты для деплоя и оркестрации (Docker, Docker Compose, системы управления конфигурациями), развертывать все компоненты системы (backend, frontend, базу данных) и обеспечивать их корректное взаимодействие в новой среде, выявлять и устранять типовые проблемы, возникающие после развертывания (логирование, анализ ошибок)

Типовые контрольные задания

Настроить CI/CD пайплайна.

Осуществить деплой приложения в облачную среду.

Создать Docker-контейнеры для приложения.

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 экзамену, зачету

Примерные вопросы для подготовки к зачёту

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)?

Примерные вопросы для подготовки к экзамену

1. Проанализируйте критерии выбора стека технологий (язык, фреймворк, БД) для нового веб-проекта. Какие факторы являются ключевыми?
2. Сравните монолитную и микросервисную архитектуры. Их достоинства, недостатки и области применения.
3. Опишите жизненный цикл HTTP-запроса в fullstack-приложении: от нажатия кнопки в интерфейсе до сохранения данных в БД и получения ответа.

4. Что такое "чистая архитектура" (Clean Architecture) и каковы её цели? Как она влияет на поддерживаемость кода?
5. Стратегии работы с состоянием (state management) в большом frontend-приложении. Когда достаточно Context API, а когда нужен Redux/MobX?
6. Детально опишите процесс настройки аутентификации и авторизации в fullstack-приложении (регистрация, логин, защита маршрутов, роли).
7. Объясните принципы обеспечения безопасности API: CORS, защита от SQL-инъекций, XSS, CSRF. Как реализовать каждую из этих защит на практике?
8. Процесс рефакторинга монолитного приложения в сторону микросервисов. Какие сложности возникают и как их решать?
9. Опишите полный CI/CD пайплайн для fullstack-приложения, от коммита кода до деплоя в облако. Какие этапы он должен включать и почему?
10. Стратегии деплоя: "синий-зелёный" (blue-green), канареечный (canary). Их преимущества, недостатки и сценарии использования.
11. Как организовать мониторинг и логирование развернутого приложения? Какие метрики и логи являются критически важными для оперативного реагирования на проблемы?
12. Объясните, как Docker и Docker Compose упрощают процесс разработки, тестирования и развертывания приложений. В чём заключаются ограничения Docker Compose и зачем тогда нужен Kubernetes?

Пример экзаменационного билета

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №

1. Теоретический вопрос (15 баллов)

Опишите процесс выбора технологического стека для разработки информационной системы. Какие критерии (технические, бизнес-человеческие) необходимо учитывать? Проиллюстрируйте свой ответ на примере гипотетического проекта (например, "Сервис для внутреннего документооборота средней компании").

2. Практико-ориентированная задача (30 баллов)

Вам необходимо спроектировать схему базы данных и API для сервиса бронирования переговорных комнат.

- Нарисуйте схему БД (основные сущности и связи) и обоснуйте выбор типа СУБД (реляционная/нереляционная).
- Спроектируйте 3 ключевых RESTful endpoint'a (укажите HTTP-метод, URL, входные и выходные данные) для этого сервиса.
- Опишите, как вы реализуете проверку прав доступа (авторизацию) при попытке пользователя отменить бронирование.

3. Ситуационный вопрос (15 баллов)

После деплоя новой версии приложения пользователи начали жаловаться на медленную загрузку страницы "Отчеты". Каков будет ваш алгоритм диагностики и решения этой проблемы? Укажите, какие инструменты (браузерные, серверные) и метрики вы будете использовать на каждом шаге.

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

Основная литература:

1. Тузовский, Анатолий Федорович Проектирование и разработка web-приложений : учебник для вузов / А. Ф. Тузовский. Электрон. дан. Москва : Юрайт, 2025 219 с (Высшее образование) URL: <https://urait.ru/bcode/561176> (дата обращения: 01.09.2025). Режим доступа: Электронно-библиотечная система Юрайт, для авториз. пользователей <https://urait.ru/bcode/561176> ISBN 978-5-534-16300-1 : 1169.00. [БИК ID: RU2fURAIT2f561176]
2. Морозова, Ольга Анатольевна Информационные технологии в государственном и муниципальном управлении : учебник для вузов / О. А. Морозова, В. В. Лосева, Л. И. Иванова. 3-е изд., пер. и доп Электрон. дан. Москва : Юрайт, 2025 156 с (Высшее образование) URL: <https://urait.ru/bcode/564458> (дата обращения: 01.09.2025). Режим доступа: Электронно-библиотечная система Юрайт, для авториз. пользователей <https://urait.ru/bcode/564458> ISBN 978-5-534-18554-6 : 719.00. [БИК ID: RU2fURAIT2f564458]

Дополнительная литература:

1. Демидов, Л.Н. Информационные технологии : Учебник / Л.Н. Демидов, В.Б. Терновсков, С.М. Григорьев, Д.В. Крахмалев Электрон. дан. Москва : КноРус, 2023 222 с. Режим доступа: book.ru Internet access <https://book.ru/book/948312> ISBN 978-5-406-11050-8. [БИК ID: RU\bookru\bibl\948312]

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>
4. •Электронно-библиотечная система «Университетская библиотека ОНЛАЙН» <http://biblioclub.ru/>
5. •Электронно-библиотечная система издательства Проспект <http://ebs.prospekt.org/books>

6. •Видеотека учебных фильмов «Решение» (тематические коллекции «Менеджмент», «Маркетинг. Коммерция. Логистика», «Юриспруденция», «Управление персоналом», «Психология управления»: <http://eduvideo.online/>

10. Методические указания для обучающихся по освоению дисциплины

Технология поэтапного освоения дисциплины

ШАГ 1: Теоретическое погружение

- Изучите презентации, лекционные материалы.
- Посмотрите 1-2 коротких видеоурока по теме.
- Составьте глоссарий из 10-15 ключевых терминов модуля.

ШАГ 2: Микропрактика

- Выполните задания на интерактивных платформах:
- HTML/CSS: [HTML Academy](#) , [CSS Grid Garden](#)
- JavaScript: [FreeCodeCamp](#) , [Codewars](#) (5-7 kyu)
- Algorithms: [LeetCode](#) (Easy задачи)
- Создайте "песочницу" (Sandbox) на CodePen или CodeSandbox для быстрых экспериментов.

ШАГ 3: Интеграционное задание

- Выполните основное практическое задание модуля (лабораторную работу).
- Важно: Сначала напишите код самостоятельно, только затем сравнивайте с эталоном или ищите решение.
- При возникновении ошибок:
 1. Прочитайте сообщение об ошибке.
 2. Изучите с помощью поисковика текст ошибки (на английском).
 3. Проверьте код через отладчик (debugger) или console.log().
 4. Обратитесь к документации (MDN Web Docs, официальная док. фреймворка).

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

Современные профессиональные базы данных и информационные справочные системы:

1. Электронная энциклопедия: <http://ru.wikipedia.org/wiki/Wiki>
2. Размещение ЭУК: <https://www.campus.fa.ru/>
3. Информационно-правовая система «Гарант»
4. Информационно-правовая система «Консультант Плюс»
5. 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)
2. **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)

3. Библиотека, читальный зал