МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ТЕХНОЛОГИЧЕСКОГО ОБРАЗОВАНИЯ
Кафедра информационных технологий и электронного обучения

Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

АНАЛИТИЧЕСКИЙ ОБЗОР

по теме: web-технологии (Web service design)

выполнил:
обучающийся 4 курса 4об_ИВТ-2/22
Спирянов Максим Дмитриевич

Санкт-Петербург
2025

ВВЕДЕНИЕ

Веб-сервисы (Web service design) представляют собой фундаментальную дисциплину цифровой эпохи, определяющую принципы создания, взаимодействия и интеграции распределенных программных систем через сетевые протоколы. Это область, где элегантность программной архитектуры встречается с жесткими требованиями масштабируемости, безопасности и пользовательского опыта. От простых API для обмена данными до сложных экосистем микросервисов, управляющих глобальными платформами, дизайн веб-сервисов эволюционировал от обслуживания машин к обслуживанию людей и бизнес-процессов, становясь ключевым драйвером инноваций и операционной эффективности в современном мире.

Современный дизайн веб-сервисов сохраняет свою основу в виде стандартизированных протоколов, таких как HTTP, и языков обмена, подобных XML и JSON, но радикально трансформируется под влиянием новых парадигм: перехода от монолитных систем к распределенным микросервисам и бессерверным функциям; требований к оркестрации сложных, сквозных пользовательских сценариев (customer journeys) across multiple channels; а также необходимости глубокой интеграции принципов безопасности и удобства использования (UX) на этапе проектирования.

В данном обзоре рассматривается эволюция технологий и архитектурных стилей веб-сервисов — от ранних стандартов SOAP и RPC до современных RESTful API, микросервисов и событийно-ориентированных архитектур. Особое внимание уделяется методологиям проектирования, ориентированным на пользователя и бизнес-процессы, что позволяет понять не только технические аспекты построения сервисов, но и их роль в создании целостного цифрового опыта в условиях тотальной связанности и автоматизации.

История развития веб-сервисов неразрывно связана с эволюцией самого интернета и бизнес-требований к программному обеспечению.

Ранняя стандартизация и основа была заложена в конце 1990-х – начале 2000-х годов. Ключевыми технологиями того периода стали XML (eXtensible Markup Language) как универсальный формат для структурированных данных, SOAP (Simple Object Access Protocol) как протокол для обмена структурированными сообщениями, WSDL (Web Services Description Language) для формального описания интерфейсов и UDDI (Universal Description, Discovery and Integration) для их реестра и поиска. Эти стандарты формировали концепцию сервисно-ориентированной архитектуры (SOA), где сервисы были сравнительно крупными и общались через централизованные шины, часто используя тяжеловесный XML.

С распространением Web 2.0 и ростом потребности в более легковесной и гибкой интеграции на первый план вышел архитектурный стиль REST (Representational State Transfer), популяризированный Роем Филдингом. RESTful API, использующие простоту HTTP-методов (GET, POST, PUT, DELETE) и форматы вроде JSON, стали де-факто стандартом для публичных и внутренних API благодаря своей простоте, производительности и лучшей приспособленности к работе с веб-клиентами.

Следующим логическим шагом, ответом на сложность поддержки и масштабирования крупных монолитных приложений, стал переход к микросервисной архитектуре. Она декомпозирует приложение на набор небольших, слабосвязанных сервисов, каждый из которых отвечает за конкретную бизнес-возможность, развертывается независимо и общается по сети, часто через легковесные механизмы вроде REST или очередей сообщений.

Сравнение ключевых архитектурных подходов выглядит следующим образом:

Аспект	**Монолитная архитектура**	**Сервисно-ориентированная архитектура (SOA)**	**Микросервисная архитектура**
Коммуникация	Внутрипроцессные вызовы (например, методы класса)	Централизованная шина (ESB), тяжелые протоколы (SOAP/WS-*)	Децентрализованная, легкие протоколы (HTTP/REST, gRPC, асинхронные сообщения)
Гранулярность	Крупное, единое приложение	Крупные, сложные сервисы	Мелкие, фокусированные на одной задаче сервисы
Гибкость и скорость изменений	Низкая: изменение требует пересборки и развертывания всего монолита	Умеренная: сервисы могут обновляться относительно независимо, но через общую шину	Высокая: каждый сервис можно разрабатывать, развертывать и масштабировать автономно
Основная сложность	Масштабирование и сопровождение растущей кодовой базы	Сложность настройки и производительность ESB, управление контрактами	Оркестрация, распределенная трассировка, управление данными в распределенной системе

Современная экосистема веб-сервисов представляет собой сложный пазл из взаимосвязанных компонентов и паттернов, обеспечивающих надежность, масштабируемость и удобство управления.

Ключевые инфраструктурные компоненты:

• API Gateway (API-шлюз) — единая точка входа для клиентов, которая выполняет маршрутизацию запросов к соответствующим микросервисам, а также отвечает за аутентификацию, ограничение частоты запросов (rate limiting), мониторинг и преобразование протоколов.

• Балансировщик нагрузки (Load Balancer) — распределяет входящий сетевой трафик между несколькими экземплярами сервиса для обеспечения отказоустойчивости и повышения производительности.

• Очереди сообщений (Message Queues) и брокеры событий — обеспечивают асинхронную, надежную коммуникацию между сервисами по паттерну «издатель-подписчик» (pub/sub), что повышает устойчивость системы к временным сбоям отдельных компонентов.

• Сети доставки контента (CDN) — географически распределенная сеть прокси-серверов для быстрой доставки статического контента (изображения, CSS, JS) пользователям по всему миру, что также помогает смягчать DDoS-атаки.

Эволюция способов исполнения кода прошла путь от физических серверов к виртуальным машинам (VM), затем к более легковесным контейнерам (например, Docker), управляемым оркестраторами (Kubernetes), и далее к бессерверным (serverless) вычислениям. В бессерверной модели разработчик развертывает отдельные функции (например, AWS Lambda, облачные функции), а облачный провайдер полностью управляет инфраструктурой, автоматически масштабируя ее в зависимости от нагрузки.

Современный дизайн веб-сервисов — это не только техническая, но и гуманитарная дисциплина. Процесс проектирования услуг (Service Design Process) ставит во главу угла потребности пользователей и эффективность бизнес-процессов.

Основные этапы этого процесса включают:

1. Исследование и обнаружение через интервью и наблюдение за пользователями для выявления реальных проблем.

2. Моделирование с помощью карт пользовательского опыта (Customer Journey Maps), которые визуализируют все взаимодействия пользователя с сервисом across multiple channels, и карт рабочих процессов (Workflow Mapping), которые детализируют внутренние бизнес-процессы.

3. Создание прототипов и итеративная разработка, часто с использованием low-code платформ для быстрой проверки гипотез.

4. Визуальное и функциональное тестирование с реальными пользователями.

5. Внедрение с использованием масштабируемых дизайн-систем для обеспечения согласованности интерфейсов.

6. Непрерывная оценка пользовательского опыта и производительности после запуска.

Для создания качественных веб-сервисов специалисты опираются на устоявшиеся лучшие практики и фундаментальную литературу.

Лучшие практики проектирования API и сервисов:

1. Четкая документация: Полная и понятная документация — залог успешной интеграции.

2. Согласованность: Последовательное именование эндпоинтов и структура URL.

3. Управление версиями: Планирование изменений API без нарушения работы существующих клиентов.

4. Безопасность: Обязательное использование HTTPS, реализация аутентификации (OAuth 2.0, JWT), валидация входящих данных и применение rate limiting.

5. Надежная обработка ошибок: Использование стандартных HTTP-кодов статусов и информативных сообщений об ошибках.

Фундаментальная литература для архитекторов и разработчиков, рекомендованная профессионалами отрасли, включает такие работы, как "Паттерны объектно-ориентированного проектирования" (банды четырех), "Микросервисы. Паттерны разработки и рефакторинга" Криса Ричардсона, а также книги, посвященные domain-driven design (DDD) и облачным технологиям.

мТаким образом, проектирование веб-сервисов сегодня — это синтез технического мастерства, архитектурного видения и глубокого понимания

человеко-компьютерного взаимодействия. Успех в этой области определяется способностью не только выстроить отказоустойчивую и масштабируемую систему, но и создать с ее помощью настоящую ценность для конечного пользователя и бизнеса.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Современные методы создания и оформления сайтов — https://astobr.com/articles/sovremennye-metody-sozdaniya-i-oformleniya-saytov/

2. Архитектура современных веб-сервисов и способы их защиты — https://innostage-group.ru/press/blog/technical/architecture-of-modern-web-services/

3. Веб-дизайн: главные тренды и инновации — https://artgluck.ru/blog/veb-dizajn-glavnyie-trendyi-i-innovaczii/

4. Best Practices for Designing Web Services — https://dev.to/ovaisnaseem/best-practices-for-designing-web-services-27ng