**Понятия сервиса, веб-сервиса, сервиc-ориентированной и микросервисной архитектуры. Поддерживающие протоколы прикладного уровня.**

**Сервис** (англ. service) - функция маркетинга, обеспечивающая реализацию услуг, связанных со сбытом, эксплуатацией продукции. Качество сервисного обслуживания повышает спрос, влияет на конкурентоспособность продукции.

*В переводе с английского сервис* ***service*** *— это просто услуга, предоставление услуги или обслуживание клиента.*

Часто данное понятие применяется к гостиничному или ресторанному бизнесу, сфере обслуживания автомобилей и даже к специализированным техническим службам, осуществляющим обслуживание какой-либо техники после ее продажи клиенту.

**Понятие "сервис" в быту и компьютерных технологиях**. Не менее часто понятие "сервис" встречается и в компьютерном мире. К примеру, в тех же «операционках» Windows (или любых других) существуют специальные сервисные программы и апплеты (компоненты, выполняющие «узкие» задачи), выполнение которых направлено на обслуживание всей системы в целом для предотвращения или устранения сбоев в работе.

Сюда же можно отнести и понятие веб-сервисов, которые предоставляют какие-либо услуги в сети Интернет при помощи специализированных программ: услуги интернет-провайдеров, электронную коммерцию в виде интернет-магазинов со службой доставки товара, разработку сайтов, интернет-приложений и т.д. [1] (<https://fb.ru/article/197614/chto-takoe-servis-harakteristika-ponyatiya>)

Всемирная паутина (World Wide Web) является готовой платформой для создания и использования распределенных машинно-ориентированных систем на основе веб-сервисов. Веб-служба, **веб-сервис** (англ. web-service) — это сетевая технология, обеспечивающая межпрограммное взаимодействие на основе веб-стандартов.

*Изначально World Wide Web была сетью документов. Web-серверы общались с клиентами по протоколу HTTP (Hypertext Transfer Protocol) и пересылали информацию в форме гипертекстовых документов, созданных средствами языка HTML (Hypertext Markup Language). Такие документы отображались в браузерах и содержали ссылки на другие документы.*

Веб-сервер выступает в качестве сервера приложений, к которым обращаются не конечные пользователи, а сторонние приложения. Это позволяет многократно использовать функциональные элементы, устранить дублирование кода, упростить решение задач интеграции приложений.

Веб - сервис - идентифицируемая веб – адресом (строка URI) программа. Программный интерфейс, представленный в машинно-обрабатываемом формате [WSDL](http://www.4stud.info/networking/web-services.html#wsdl). Другие системы взаимодействуют с этим веб-сервисом путем обмена сообщениями протокола [SOAP](http://www.4stud.info/networking/web-services.html#soap). В качестве транспорта для сообщений используется протокол [HTTP](http://www.4stud.info/web-programming/protocol-http.html). Описание веб-сервисов и их API могут быть найдены средствами [UDDI](http://www.4stud.info/networking/web-services.html#uddi). Концепция веб-сервиса приведена на [рис. 1.](http://www.4stud.info/networking/web-services.html#web-service-idea), а связь между протоколами — на рис. 2.

*В обиходе веб - сервисами называют услуги, оказываемые в Интернете. В этом употреблении термин требует уточнения, идёт ли речь о поиске, веб - почте, хранении документов, файлов, закладок. Веб - сервисами можно пользоваться независимо от компьютера, браузера или места доступа в Интернет.*

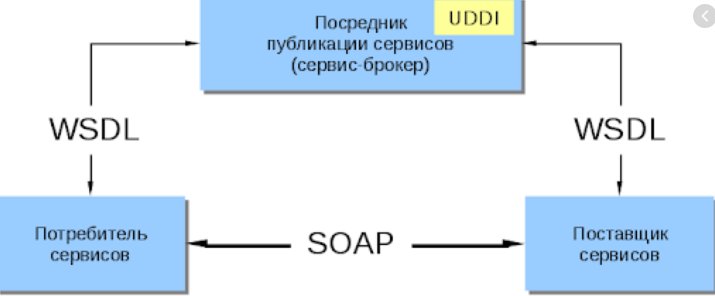


Рис. 1 Концепция веб-сервиса

**SOAP** (Simple Object Access Protocol) — протокол обмена сообщениями между потребителем и поставщиком веб-сервиса;

**WSDL** (Web Services Description Language) — язык описания внешних интерфейсов веб-службы;

**UDDI**(Universal Discovery, Description and Integration) — универсальный интерфейс распознавания, описания и интеграции, используемый для формирования каталога веб-сервисов и доступа к нему.



Рис.2 Связь между протоколами

Все спецификации, используемые в технологии, основаны на [XML](http://www.4stud.info/web-programming/xml.html) и, соответственно, наследуют его преимущества (структурированность, гибкость и т.д.) и недостатки (громоздкость, медлительность)[2](<http://www.4stud.info/networking/web-services.html>).

*В веб-сервисах всегда есть клиент и сервер. Сервер – это и есть веб-сервис и иногда его называют endpoint (конечная точка, куда доходят SOAP сообщения от клиента).*

Примером веб-сервиса может стать компания Amazon. Организация имеет сеть онлайн-магазинов и систем доставок. Она предоставляет веб-сервис, который запрашивает цены на продукты, продаваемые онлайн через [amazon.com](http://amazon.com/).

**РЕЗЮМЕ**

Основным компонентом веб-сервисов в интернете являются данные, которые передаются между клиентом и сервером. Веб-сервис – это система, доступная в интернет-пространстве и работающая на основе специальной программы, идентификация которой выполняется с помощью URL-строки. Поиск осуществляется другими ресурсами, основной задачей является взаимодействие программных систем на разных платформах, для чего используются открытые протоколы. К системам \*\*веб-сервис\*\* относят поисковики, хостинги, электронную почту, облачные хранилища, календари и прочие сервисы.

Ключевая особенность системы – отсутствие зависимости от характеристик и состояния какого-либо конкретного компьютера, браузера или провайдера, поэтому доступ к таким сервисам поддерживается в любом государстве. Единственное условие для пользования системой – наличие подключения к интернету.

**Понятия сервиc-ориентированной и микросервисной архитектуры**

**Архитектура системы –** принципиальная организация системы, воплощенная в её элементах, их взаимоотношения друг с другом и средой, а также принципов, направляющих проектирование архитектуры, её создание и эволюцию.

**Сервис-ориентированная архитектура** (сокращенно SOA) - архитектурный стиль, который поддерживает ориентированность на службы и является парадигмой для бизнеса и ИТ. Данный архитектурный стиль предназначен для разработки систем с точки зрения служб, доступных через интерфейс, и результатов действий этих служб. Служба - логическое представление ряда действий, который имеет определенные результаты, является автономным, может быть составлен из других служб и является "черным ящиком" для потребителей службы.  
*[\*](* [*http://docs.cntd.ru/document/1200146803*](http://docs.cntd.ru/document/1200146803)*).*

**Пример.** Представим классический интернет-магазин. Стандартные модули: UI, бизнес-логика и дата-слой. Возможны способы взаимодействия с сервисом: API REST и веб-интерфейс. **При построении монолита все эти вещи будут управляться внутри одного и того же модуля.**

**Микросервисная архитектура** (Micro Service Architecture, MSA) – принципиальная организация системы на основе **микросервисов** и их взаимодействия друг с другом и со средой по сети. Микросервисы — это шаблон сервис-ориентированной архитектуры, в котором приложения создаются как совокупность различных наименьших независимых сервисных единиц. Это программный подход, который фокусируется на разложении приложения на однофункциональные модули с четко определенными интерфейсами.

*Эти модули могут быть независимо развернуты и эксплуатироваться небольшими группами, владеющими всем жизненным циклом службы.*

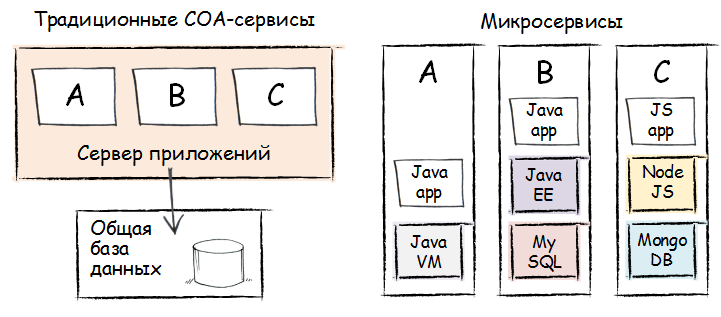
Одно из важнейших свойств микросервиса это его **независимость** от других компонентов приложения**.** Суть микросервиса по отношению к большому приложению (монолиту) изображена на рис.3.****

Рис.3 Микросервисы по отношению к монолиту

Разбиение на независимые компоненты даёт безусловные и неоспоримые преимущества: легкое понимание контекста, гибкость развития, управления и масштабирования.

Но что такого плохого в существующих архитектурных решениях, таких как SOA? В отличии от монолитных архитектурных приложений, микросервисы [3](https://javarush.ru/groups/posts/2015-mikroservisnaja-arhhitektura-pljusih-i-minusih):

* Улучшает изоляцию сбоя компонентов: большие приложения могут продолжить эффективно работать, даже при неисправности какого-то отдельного модуля.
* Устраняет приверженность приложения к одному технологическому стеку: если хочешь попробовать новый технологический стек на каком-то сервисе - пожалуйста. Зависимости будут гораздо легче, чем при монолитном, к тому же будет намного проще откатить все вспять. Чем меньше кода в одном приложении, тем легче работать.
* Упрощает понимание функционального сервиса для новых сотрудников.

**Пример.** Представим тот же классический интернет-магазин. Здесь отличие от монолита в том, что у каждого модуля (UI, бизнес-логика и дата-слой) есть свой сервис и своя БД. Они слабо связаны и могут взаимодействовать с различными протоколами (например, REST) через свои границы.

**РЕЗЮМЕ**

**Сервис-ориентированная архитектура** — это стиль проектирования ПО. Архитектура делится на две части: функциональные аспекты и аспекты качества обслуживания.

*Использование архитектурного стиля SOA может помочь ИС гибко и быстро реагировать на постоянно меняющиеся потребности бизнеса.*

**Миксросервисная архитектура** - стиль разработки, который позволяет создавать приложения в виде набора небольших автономных сервисов, разработанных для бизнес-сферы.

*Микросервисы в значительной степени получили свое название из-за того, что сервисы здесь меньше, чем в монолитной среде. Тем не менее, микро –о бизнес-возможностях, а не о размере.*

Разница между двумя системами в том, что первая представляет собой единый большой блок - монолит. Второй - это набор небольших специфических сервисов (служб). У каждой службы своя роль. Помимо децентрализации решений по концептуальным моделям, микросервисы также децентрализует решения по хранению данных. В то время как монолитные приложения предпочитают единую логическую базу данных для постоянных данных, предприятия часто предпочитают единую базу данных для целого ряда приложений.

Многое зависит от собственной организационной структуры. Есть 6 команд, которые буду работать над продуктом? Микросервисная архмтектура может подойти. У вас команды из 3-х разработчиклв? Вероятно, они будут хорошо строить и поддерживать монолит.

Другими факторами являются скорость изменения и сложность. Высокоие темпы изменений и высокая сложность могут быть факторами, заставляющими выбрать архитектуру микросервиса.

[Сравнение архитектур]( <https://coderlessons.com/tutorials/veb-razrabotka/arkhitektura-veb-servisov/11-soa-protiv-mikroservisov>).

**Поддерживающие протоколы прикладного уровня**

[**Прикладной уровень**](https://zvondozvon.ru/tehnologii/prikladnoi-uroven-osi)**(APPLICATION).** Необходим для взаимодействия между собой сетевых приложений, таких как web, e-mail, skype и тд.

*Задачей данного уровня является обеспечение доступа к сетевым службам.*

**Функции прикладного уровня:**

* Решение задач, отправка файлов; управление заданиями и системой;
* Определение пользователей по их логину, e-mail адресу, паролям, электронным подписям;
* Запросы на соединение с иными прикладными процессами;

В модели [TCP/IP](<https://sites.google.com/site/gosyvmkss12/ceti-ebm-i-cpedctva-telekommynikacij/16-stek-tsr-ip-protokoly-prikladnogo-urovna>) прикладной уровень это единственный уровень, который находится выше транспортного. А в модели взаимодействия открытых систем ([OSI](https://zvondozvon.ru/tehnologii/model-osi)) есть еще 2 уровня: уровень представления и сеансовый. Наиболее известные протоколы прикладного уровня на рис.4-5.

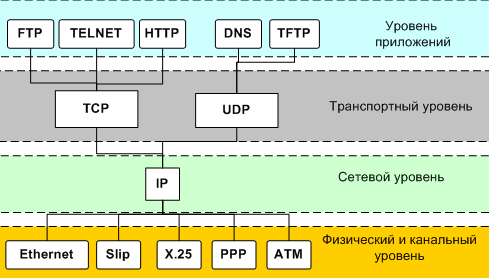
****

Рис.4 Протоколы стека TCP/IP

*Почему существуют два транспортных протокола TCP и UDP, а не один из них? Дело в том, что они предоставляют разные услуги прикладным процессам. [4]( https://refdb.ru/look/2224218-pall.html).*



Рис.5. Протоколы стека TCP/IP прикладного уровня

Самый верхний уровень в иерархии протоколов Интернет занимают следующие **протоколы прикладного уровня** [3]( http://enisey.name/umk/ipr/ch01s07.html).:

* DNS - распределенная система доменных имен, которая по запросу, содержащему доменное имя хоста сообщает IP адрес;
* HTTP - протокол передачи гипертекста в Интернет;
* HTTPS - расширение протокола HTTP, поддерживающее шифрование;
* FTP - протокол, предназначенный для передачи файлов в компьютерных сетях;
* Telnet - сетевой протокол для реализации текстового интерфейса по сети;
* SSH - протокол прикладного уровня, позволяющий производить удаленное управление операционной системой и передачу файлов. В отличие от Telnet шифрует весь трафик;
* POP3 – протокол почтового клиента, который используется почтовым клиентом для получения сообщений электронной почты с сервера;
* IMAP - протокол доступа к электронной почте в Интернет;
* SMTP – протокол, который используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю;
* LDAP - протокол для доступа к службе каталогов X.500, является широко используемым стандартом доступа к службам каталогов;
* XMPP - основанный на XML расширяемый протокол для мгновенного обмена сообщениями в почти реальном времени;
* SNMP - базовый протокол управления сети Internet.

**Более подробно некоторые из этих протоколов**

Протокол **Telnet** – это прикладной протокол стека TCP/IP, обеспечивающий эмуляцию терминалов. Терминал – это устройство, состоящее из монитора и клавиатуры и используемое для взаимодействия с хост- компьютерами на которых выполняются программы. Программы запускаются на хосте, поскольку терминалы, как правило, не имеют собственного процессора.

Secure Shell (**SSH**) — сетевой протокол, позволяющий производить удалённое управление компьютером и передачу файлов. Сходен по функциональности с протоколом Telnet и rlogin, однако использует алгоритмы шифрования передаваемой информации

File Transfer Protocol **(FTP)** – это приложение, позволяющее с помощью протокола TCP передать данные от одного удаленного устройства к другому. Как и в протоколе Telnet, заголовок FTP и соответствующие данные инкапсулируются в поле полезной нагрузки пакета TCP. FTP поддерживает передачу двоичных или текстовых (ASCII) файлов. FTP предназначен для передачи файлов целиком, что делает его удобным средством для пересылки через глобальную сеть файлов большого размера FTP не позволяет передать часть файла или некоторые записи внутри файла.

Протокол Simple Mail Transfer Protocol (**SMTP**) предназначен для передачи сообщений электронной почты между сетевыми системами. С помощью этого протокола системы UNIX, OpenVMS, Windows и Novell NetWare могут пересылать электронную почту поверх протокола TCP. SMTP можно рассматривать как альтернативу протоколу FTP при передаче файла от одного компьютера к другому. При работе с SMTP не нужно знать имя учетной записи и пароль для удаленной системы. Все, что нужно, – это адрес электронной почты принимающего узла. SMTP может пересылать только текстовые файлы, поэтому файлы в других форматах должны быть конвертированы в текстовый вид, только после этого их можно поместить в SМТР-сообщение.

Domain Name System **(DNS -** служба имен доменов**)** представляет собой службу стека TCP/IP, преобразующую имя компьютера или домена в IP-адрес или, наоборот, конвертирующую IP-адрес в компьютерное или доменное имя. Этот процесс называется разрешением (имен или адресов). Пользователям легче запоминать имена, а не IP-адреса в десятичном представлении с разделительными точками, однако поскольку компьютерам все равно нужны IP-адреса, то должен быть способ преобразования одного способа адресации в другой. Для этого служба DNS использует таблицы просмотра, в которых хранятся пары соответствующих значений.

[4](https://sites.google.com/site/gosyvmkss12/ceti-ebm-i-cpedctva-telekommynikacij/16-stek-tsr-ip-protokoly-prikladnogo-urovna).