5. **Service-Oriented Architecture and Web Services**

Service-Oriented Architecture (SOA) е етап в еволюцията на разработката и/или интеграцията на приложения. Тя определя начин за създаване на преизползваеми софтуерни компоненти чрез интерфейси. SOA е архитектурен подход, при който приложенията използват услуги, достъпни в мрежата. В тази архитектура услугите се предоставят за изграждане на приложения чрез мрежово извикване (network call) през интернет. Използват се стандартизирани комуникационни протоколи, за да се ускори и опрости интеграцията на услугите в приложенията. Всяка услуга в SOA е пълна бизнес-функция сама по себе си. Услугите се публикуват така, че разработчиците лесно да могат да ги използват при създаване на приложения. SOA не е **microservice architecture**.

Основни характеристики на SOA:

- интероперативност (interoperability, оперативна съвместимост) между услугите.
- методи за инкапсулация на услуги (service encapsulation), откриване на услуги (service discovery), композиция на услуги (service composition), преизползваемост (reusability) и интеграция (integration).
- QoS (Quality of Services) чрез service contract, базиран на Service Level Agreement (SLA).
- loose coupling – слабо свързани услуги.
- location transparency – прозрачност на местоположението, с по-добра мащабируемост и наличност.
- Улеснява поддръжката и намалява разходите за разработка и внедряване на приложения.

Основни роли в SOA:

- **Service provider** (доставчик на услуги): Поддържа услугата и я прави достъпна за други. За да популяризира услугите си, доставчикът може да ги публикува в registry (регистър), заедно със service contract, който описва естеството на услугата, как да се използва, изискванията и евентуалните такси.
- **Service consumer** (потребител на услуги): Може да открие метаданните (metadata) на услугата в регистъра и да разработи клиентски компоненти за връзка и използване на услугата.

Orchestration -Услугите могат да **обединяват информация от други услуги** или да създават **workflows за задоволяване на нуждите на потребителя**.

Service choreography – координирано взаимодействие между услуги без централен контрол.

Компоненти на SOA:

- **Standardized service contract**: Описан чрез един или няколко service description documents.
- **Loose coupling**: Услугите са самостоятелни компоненти с минимална зависимост от други.
- **Abstraction**: Всяка услуга се дефинира чрез контракти и описания, които скриват нейната вътрешна логика.
- **Reusability**: Услугите са проектирани така, че да могат да се преизползват, спестявайки време и разходи.
- **Autonomy**: Услугите контролират собствената си логика, без потребителят да знае как е реализирана.
- **Discoverability**: Чрез description documents и metadata услугите могат да бъдат откривани.

- **Composability**: Услугите могат да се комбинират за реализиране на сложни операции чрез orchestration и choreography.

Предимства на SOA:

- **Service reusability**: Приложенията се създават от съществуващи услуги, което улеснява преизползването.
- **Easy maintenance**: Тъй като услугите са независими, те могат да бъдат модифицирани без да влияят на други.
- **Platform independent**: Позволява изграждане на приложения чрез услуги от различни платформи.
- **Availability**: Услугите могат да бъдат достъпни при поискване.
- **Reliability**: Приложенията са по-надеждни, тъй като по-малките услуги се откриват и отстраняват по-лесно.
- **Scalability**: Услугите могат да се изпълняват на различни сървъри, което подобрява мащабируемостта.

Недостатъци на SOA:

- **High overhead**: Валидирането на параметрите при всяка интеракция намалява производителността.
- **High investment:** Значителна първоначална инвестиция.
- **Complex service management:** Когато услугите обменят милиони съобщения, става трудно за управление.

Web service е стандартизиран метод за предаване на съобщения между клиентски и сървърни приложения. **Web service** е софтуерен модул, предназначен да изпълнява определен набор от функции. В **cloud computing**, **web services** могат да бъдат открити и извикани през мрежата. **Web service** има способността да доставя функционалност на клиента, който го е извикал.

Web service представлява набор от отворени протоколи и стандарти, които позволяват обмен на данни между различни приложения или системи. **Web services** могат да бъдат използвани от софтуерни програми, написани на различни програмни езици и работещи на различни платформи, за да обменят данни през компютърни мрежи като Интернет.

C**loud** технология, която използва стандартизирани **web protocols** (**HTTP**, **HTTPS**) за свързването и обмен на съобщения – обикновено във формат **XML (Extensible Markup Language)** – се счита за **web service**. W**eb services** позволяват на програми, разработени на различни езици, да комуникират помежду си чрез обмен на данни между **client** и **server**. Клиентът извиква **web service**, като подава **XML** заявка, на която се отговаря с **XML** отговор.

Функции на Web Services: Достъпни са чрез интернет или вътрешна мрежа (**intranet**); Стандартизиран **XML messaging protocol;** Независими от операционна система или език за програмиране; Самоописващи се чрез **XML;** Могат да бъдат лесно открити чрез локационен механизъм

Компоненти на Web Service - Най-основната платформа за **web services** включва **XML** и **HTTP**. Всички типични **web services** използват следните компоненти:

SOAP (Simple Object Access Protocol) - протокол за съобщения, независим от транспортния механизъм. Базиран е на изпращане на **XML** данни под формата на **SOAP messages**. Всеки SOAP документ съдържа **root element**, наречен *envelope*, разделен на две части – **header** и **body**. **Header** съдържа информация за маршрутизиране – към кой клиент да се изпрати съобщението; **Body** съдържа основното съобщение

UDDI (Universal Description, Discovery, and Integration) - стандарт за описване, публикуване и откриване на **web services**. Той предоставя хранилище за **WSDL** файлове, които клиентските приложения могат да използват, за да научат какви действия предоставя съответната услуга. Може да се сравни с телефонен указател – съдържа име, адрес и други данни за услугата.

WSDL (Web Services Description Language) - Когато една **web service** не може да бъде открита – тя не може и да бъде използвана. Клиентът трябва да знае: Къде се намира услугата; Какво прави тя

WSDL - XML-базиран файл, описващ какви действия изпълнява **web service** и как клиентът може да взаимодейства с нея.

Как работи една Web Service? Клиентът изпраща поредица от **web service calls** до сървъра, който хоства реалната услуга. Тези заявки се извършват чрез **Remote Procedure Calls (RPC)**.

Пример: **Flipkart** предлага **web service**, която показва цените на продуктите. **Frontend** може да е написан на .NET или Java, но **web service** работи независимо от езика.

Най-важната част е обменът на **XML** данни – междинен език, разбираем от различни програмни езици. Тези данни се предават чрез **SOAP** и стандартния **HTTP** протокол.

Характеристики на Web Services

(a) XML Based - Използва **XML** за представяне на данни – независим от платформа и ОС. Предоставя висока **interoperability**.

(b) Loosely Coupled - Клиентът не е силно обвързан с доставчика. Промени в интерфейса на услугата не изискват задължително промени при клиента.

(c) Synchronous or Asynchronous -Synchronous: клиентът чака изпълнението; **Asynchronous**: клиентът продължава други задачи и получава резултат по-късно

(d) Coarse-Grained - Вместо фини (fine-grained) методи, **web services** предлагат по-груби (coarse-grained) операции, обединяващи няколко действия.

(e) Supports Remote Procedural Call - Позволява извикване на функции на отдалечени обекти. Поддържа технологии като **EJB**, **.NET**, и други **RPC** механизми.

(f) Supports Document Exchanges - Позволява лесен обмен на цели **documents** – от прости адреси до сложни заявки (напр. **Request for Quotation**).

Предимства на Web Services

(a) Business Functions over the Internet - Web service може да бъде достъпен отвсякъде чрез **HTTP**, което го прави универсално използваем.

(b) Interoperability - Позволява комуникация между приложения, написани на различни езици – напр. .NET и Java.

(c) Low Cost Communication - Използването на **SOAP over HTTP** прави внедряването евтино чрез съществуваща интернет връзка.

(d) Standard Protocol - Комуникацията е базирана на индустриален стандарт – с четири слоя: **Service Transport**, **XML Messaging**, **Service Description**, **Service Discovery**.

(e) Reusability - Една **web service** може да бъде използвана от множество клиентски приложения едновременно.