Servicios Web y microservicios

Javier Albert Segui

Curso 2020 / 21

Universidad de Alcalá

Servicios Web

- Del inglés, Web Services.
- Es una **tecnología** que utiliza un conjunto de **protocolos** y **estándares** que sirven para **intercambiar datos** entre aplicaciones.
- La interoperabilidad entre distintos sistemas, lenguajes y plataformas se consigue mediante la adopción de unos estándares abiertos.

Servicios Web

Según el W3C un servicio web es:

"Un sistema software diseñado para soportar la interacción M2M, a través de una red, de forma interoperable."

Cuenta con una interfaz descrita en un formato procesable por un equipo informático (específicamente en WSDL), a través de la que es posible interactuar con el mismo mediante el intercambio de mensajes SOAP, típicamente transmitidos usando serialización XML sobre HTTP conjuntamente con otros estándares web.

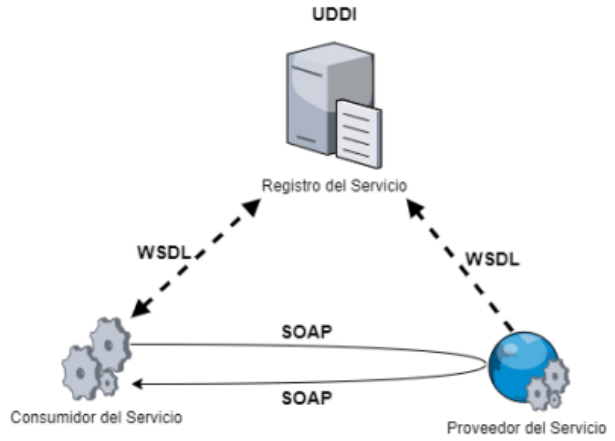
Servicios Web

- Las características deseables de un Servicio Web son:
  - **Un servicio debe poder ser accesible a través de la Web**. Para ello debe utilizar protocolos de transporte estándares como HTTP, y codificar los mensajes en un lenguaje estándar que pueda conocer cualquier cliente que quiera utilizar el servicio.
  - **Un servicio debe contener una descripción de sí mismo**. De esta forma, una aplicación podrá saber cuál es la función de un determinado Servicio Web, y cuál es su interfaz, de manera que pueda ser utilizado de forma automática por cualquier aplicación, sin la intervención del usuario.
  - **Debe poder ser localizado**. Deberemos tener algún mecanismo que nos permita encontrar un Servicio Web que realice una determinada función. De esta forma tendremos la posibilidad de que una aplicación localice el servicio que necesite de forma automática, sin tener que conocerlo previamente el usuario.

Servicios Web - Arquitectura

- Arquitectura funcional: Podemos distinguir 3 agentes distintos en funciones.

Proveedor de servicio	Implementa unas determinadas operaciones (servicio). Un cliente podrá solicitar uno de estos servicios a este proveedor.
Cliente del servicio	Invoca a un proveedor de servicio para la realización de alguna de las operaciones que proporciona.
Registro de servicios	Mantiene una lista de proveedores de servicios disponibles, junto a sus descripciones.



Servicios Web - Arquitectura

- Arquitectura de capas de protocolos:

Capa	Descripción
Transporte de servicios	Es la capa que se encarga de transportar los mensajes entre aplicaciones. Normalmente se utiliza el protocolo HTTP para este transporte, aunque los servicios web pueden viajar mediante otros protocolos de transferencia de hipertexto como SMTP, FTP o BEEP.
Mensajería XML	Es la capa responsable de codificar los mensajes en XML de forma que puedan ser entendidos por cualquier aplicación. Puede implementar los protocolos XML-RPC o SOAP.
Descripción de servicios	Se encarga de definir la interfaz pública de un determinado servicio. Está definición se realiza mediante WSDL.
Localización de servicios	Se encarga del registro centralizado de servicios, permitiendo que estos sean anunciados y localizados. Para ello se utiliza el protocolo UDDI.

Servicios Web

- Hay 2 tipos de Servicios Web:
  - SOAP
  - RESTful

Servicios Web – SOAP

- SOAP – Simple Object Access Protocol
  - Es un protocolo escrito en XML para el intercambio de información entre aplicaciones. Es un formato para enviar mensajes, diseñado especialmente para servir de comunicación en Internet, pudiendo extender los HTTP headers. Es una forma de definir qué información se envía y cómo mediante XML.
  - Dos tipos de mensajes:
    - Mensajes orientados al documento: Contienen cualquier tipo de contenido que queramos enviar entre aplicaciones.
    - Mensajes orientados a RPC: Este tipo de mensajes servirá para invocar procedimientos de forma remota (Remote Procedure Calls). Podemos verlo como un tipo más concreto dentro del tipo anterior, ya que en este caso como contenido del mensaje especificaremos el método que queremos invocar junto a los parámetros que le pasamos, y el servidor nos deberá devolver como respuesta un mensaje SOAP con el resultado de invocar el método.

Servicios Web – SOAP


```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
                <ns:getTemperatura xmlns:ns="http://rvg.uah.es/ns">
                        <area>Alcala de Henares</area>
                </ns:getTemperatura>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Servicios Web – SOAP

- WSDL - Web Services Description Language
  - Es un lenguaje basado en XML para describir los servicios web y cómo acceder a ellos. Es el formato estándar para describir un web service, y fue diseñado por Microsoft e IBM. WSDL es una parte integral del estándar UDDI, y es el lenguaje que éste utiliza.
  - El fichero WSDL describirá la interfaz del Servicio Web, con los métodos a los que podemos invocar, los parámetros que debemos proporcionarles y los tipos de datos de dichos parámetros.

Servicios Web – SOAP

- El elemento raíz dentro de este fichero es **definitions**, donde se especifican los espacios de nombres que utilizamos en nuestro servicio. Dentro de este elemento raíz encontramos los siguientes elementos:
  - **types:** Se utiliza para definir los tipos de datos que se intercambiarán en el mensaje.
  - **message:** Define los distintos mensajes que se intercambiaran durante el proceso de invocación del servicio. Se deberán definir los mensajes de entrada y salida para cada operación que ofrezca el servicio. En el caso de mensajes RPC, en el mensaje de entrada se definirán los tipos de parámetros que se proporcionan, y en el de salida el tipo del valor devuelto.
  - **portType:** Define las operaciones que ofrece el servicio. De cada operación indica cuales son los mensajes de entrada y salida, de entre los mensajes definidos en el apartado anterior.
  - **binding:** Indica el protocolo y el formato de los datos para cada mensaje de los definidos anteriormente. Este formato puede ser orientado al documento u orientado a RPC. Si es orientado al documento tanto el mensaje de entrada como el de salida contendrán un documento XML. Si es orientado a RPC el mensaje de entrada contendrá el método invocado y sus parámetros, y el de salida el resultado de invocar dicho método.
  - **service:** Define el servicio como una colección de puertos a los que se puede acceder. Un puerto es la dirección (URL) donde el servicio actúa. Esta será la dirección a la que las aplicaciones deberán conectarse para acceder al servicio. Además, contiene la documentación en lenguaje natural del servicio.

Servicios Web – SOAP

```
<?xml version="1.0" encoding="utf-8" ?>
  <definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:tns="http://rvg.ua.es/wsdl"
     xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
        targetNamespace="http://rvg.ua.es/wsdl"
     xmlns="http://schemas.xmlsoap.org/wsdl/">
   <message name="getTempRequest">
          <part name="string_1"
                       xmlns:partns="http://www.w3.org/2001/XMLSchema"
                  type="partns:string" />
   </message>
   <message name="getTempResponse">
          <part name="double_1"
                       xmlns:partns="http://www.w3.org/2001/XMLSchema"
                  type="partns:double" />
   </message>
   <portType name="TempPortType">
          <operation name="getTemp">
                 <input message="tns:getTempRequest" />
                 <output message="tns:getTempResponse" />
          </operation>
   </portType>
   <binding name="TempPortSoapBinding" type="tns:TempPortType">
          <soap:binding style="rpc"
               transport="http://schemas.xmlsoap.org/soap/http" />
      <operation name="getTemp">
                       <soap:operation soapAction="" style="rpc" />
                       <input>
                              <soap:body use="encoded"
                              namespace="http://rvg.ua.es/wsdl"
                              encodingStyle=
                                     "http://schemas.xmlsoap.org/soap/encoding/" />
                       </input>
                       <output>
                              <soap:body use="encoded"
                              namespace="http://rvg.ua.es/wsdl"
                              encodingStyle=
                                     "http://schemas.xmlsoap.org/soap/encoding/" />
                          </output>
               </operation>
   </binding>
   <service name="Temp">
          <documentation>Documentacion</documentation>
          <port name="TempPort" binding="tns:TempPortSoapBinding">
                 <soap:address
                              location="http://localhost:7001/sw_temp/Temp" />
          </port>
   </service>
</definitions>
```

Servicios Web – SOAP

- UDDI - Universal Description, Discovery and Integration
  - Es un estándar XML para describir, publicar y encontrar servicios web. Es un directorio donde las compañías pueden registrar y buscar servicios web. Es un directorio de interfaces de servicios web descritos en WSDL que se comunican mediante SOAP.

Servicios Web – REST

- RESTful: son un tipo de Servicios web que se adhieren a una serie de restricciones arquitectónicas englobadas bajo REST.
- REST: proviene de la tesis doctoral de Roy Fielding, publicada en el año 2000,y significa **REpresentational State Transfer**
- REST es un conjunto de restricciones que, cuando son aplicadas al diseño de un sistema, crean un estilo arquitectónico de software.

Servicios Web – REST

- Debe ser un sistema cliente-servidor
- **Tiene que ser sin estado**, es decir, no hay necesidad de que los servicios guarden las sesiones de los usuarios (cada petición al servicio tiene que ser independiente de las demás)
- **Debe soportar un sistema de cachés**: la infraestructura de la red debería soportar caché en diferentes niveles
- **Debe ser un sistema uniformemente accesible** (con una interfaz uniforme): Esta restricción define cómo debe ser la interfaz entre clientes y servidores. La idea es simplificar y desacoplar la arquitectura, permitiendo que cada una de sus partes puede evolucionar de forma independiente
- **Tiene que ser un sistema por capas**: un cliente no puede "discernir" si está accediendo directamente al servidor, o a algún intermediario. Las "capas" intermedias van a permitir soportar la escalabilidad, así como reforzar las políticas de seguridad

Servicios Web – REST

- Un **recurso REST** es cualquier cosa que sea direccionable (y, por lo tanto, accesible) a través de la Web. Por direccionable nos referimos a recursos que puedan ser accedidos y transferidos entre clientes y servidores.
- La **representación de los recursos** es lo que se envía entre los servidores y clientes. Una representación muestra el estado temporal del dato real almacenado en algún dispositivo de almacenamiento en el momento de la petición
- **Direccionabilidad de los recursos**: URI, o Uniform Resource Identifier, en un servicio web RESTful es un hiperenlace a un recurso, y es la única forma de intercambiar representaciones entre clientes y servidores. Un servicio web RESTful expone un conjunto de recursos que identifican los objetivos de la interacción con sus clientes.

Servicios Web - REST

- Ventajas:
  - Separación entre Cliente y Servidor
  - Visibilidad, fiabilidad y escalabilidad
  - Independencia del lenguaje y Plataforma
  - Experiencia de usuario
  - Menor consumo de Recursos en el servidor
- Desventajas:
  - Complejidad aumentada
  - Centralización de la información

Servicios Web - SOAP vs REST

Ventajas REST	Ventajas SOAP
·Pocas operaciones con muchos recursos	·Muchas operaciones con pocos recursos
·Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia	·Se centra en el diseño de aplicaciones distribuidas
·HTTP GET, HTTP POST, HTTP PUT, HTTP DEL	·SMTP,HTTP POST, MQ
·XML auto descriptivo	·Tipado fuerte, XML Schema
·Síncrono	·Síncrono y Asíncrono
·HTTPS	·WS SECURITY
·Comunicación punto a punto y segura	·Comunicación origen a destino seguro

Servicios Web - Seguridad

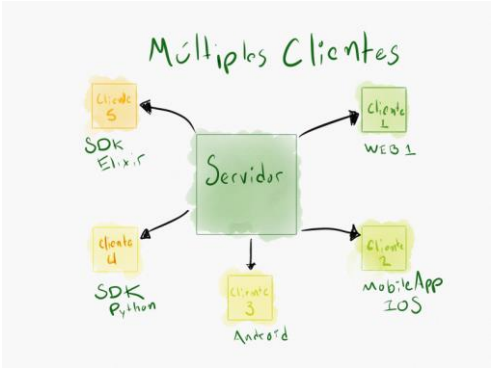
- En la utilización de los Servicios Web, encontramos problemas de seguridad en diferentes aspectos. Podemos encontrar problemas de seguridad en cuanto a la confidencialidad, la autentificación y la seguridad de la red.
- Confidencialidad:
  - Cuando un cliente utiliza un Servicio Web, deberá enviarle un mensaje a este servicio a través de la red, y el servicio le responderá mediante otro mensaje. Estos mensajes contendrán información que puede ser confidencial.
  - Dado que estos mensajes se envían mediante protocolo HTTP, podrán ser encriptados mediante SSL evitando de esta forma que puedan ser interceptados por un tercero.
- Autentificación: Puede que necesitemos identificar a un usuario para prestarle un determinado servicio, o bien para saber si tiene autorización para acceder a dicho servicio.
  - Podemos utilizar para ello la autentificación que nos proporciona el protocolo HTTP. Encontramos el mismo problema que en el caso anterior, necesitamos invocar un conjunto de servicios, deberemos autentificarnos por separado para cada uno de ellos, ya que pueden estar distribuidos en distintos servidores a través de Internet. Para solucionar este problema, deberíamos contar con un contexto compartido global de donde cualquier servicio pudiese obtener esta información de autentificación (p.e. → SSO).
- Seguridad de la red: Hemos de pensar que estamos permitiendo invocar procedimientos remotos mediante protocolo HTTP, que en un principio fue diseñado para la extracción de documentos. Por lo tanto, sus puertos no suelen ser cortados por ningún firewall, de forma cualquiera podrá utilizar estos servicios libremente, sin que los firewalls puedan controlarlo.

API

- Es una abreviatura de **Application Programming Interfaces**, que en español significa **interfaz de programación de aplicaciones**.
- Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, **permitiendo la comunicación entre dos aplicaciones** de software a través de un conjunto de reglas.
- Podemos encontrar estas API en:
  - Navegador: Geolocalización, Storage, WebWorkers….
  - Servidor: nos dan servicios de terceros de una forma sencilla para nuestras aplicaciones.

API

- Actúa como intermediario virtual, remite información de una interfaz, como una aplicación móvil, a otra. Las API conectan diferentes partes de una plataforma de software con el fin de garantizar que la información acabe en el lugar adecuado.
- Estos puntos de conexión no solo funcionan como canales para las comunicaciones internas, sino también como un modo para que las herramientas externas accedan a la misma información. Así pues, las API pueden incluirse en una de las dos categorías:
  - API privado
  - API abierto



API

- API privadas:
  - Solo pueden acceder los desarrolladores y los usuarios de la organización. Estas API normalmente conectan procesos internos de los equipos con el fin de reducir el trabajo aislado y mejorar la colaboración.
- API Abiertas
  - proporcionan a los desarrolladores externos un modo de acceder fácilmente a la información e integrarla entre herramientas. Una API abierta o pública ahorra a los desarrolladores tiempo, pues les permite conectar su plataforma con herramientas que ya tienen, lo que reduce la necesidad de crear funciones totalmente nuevas.

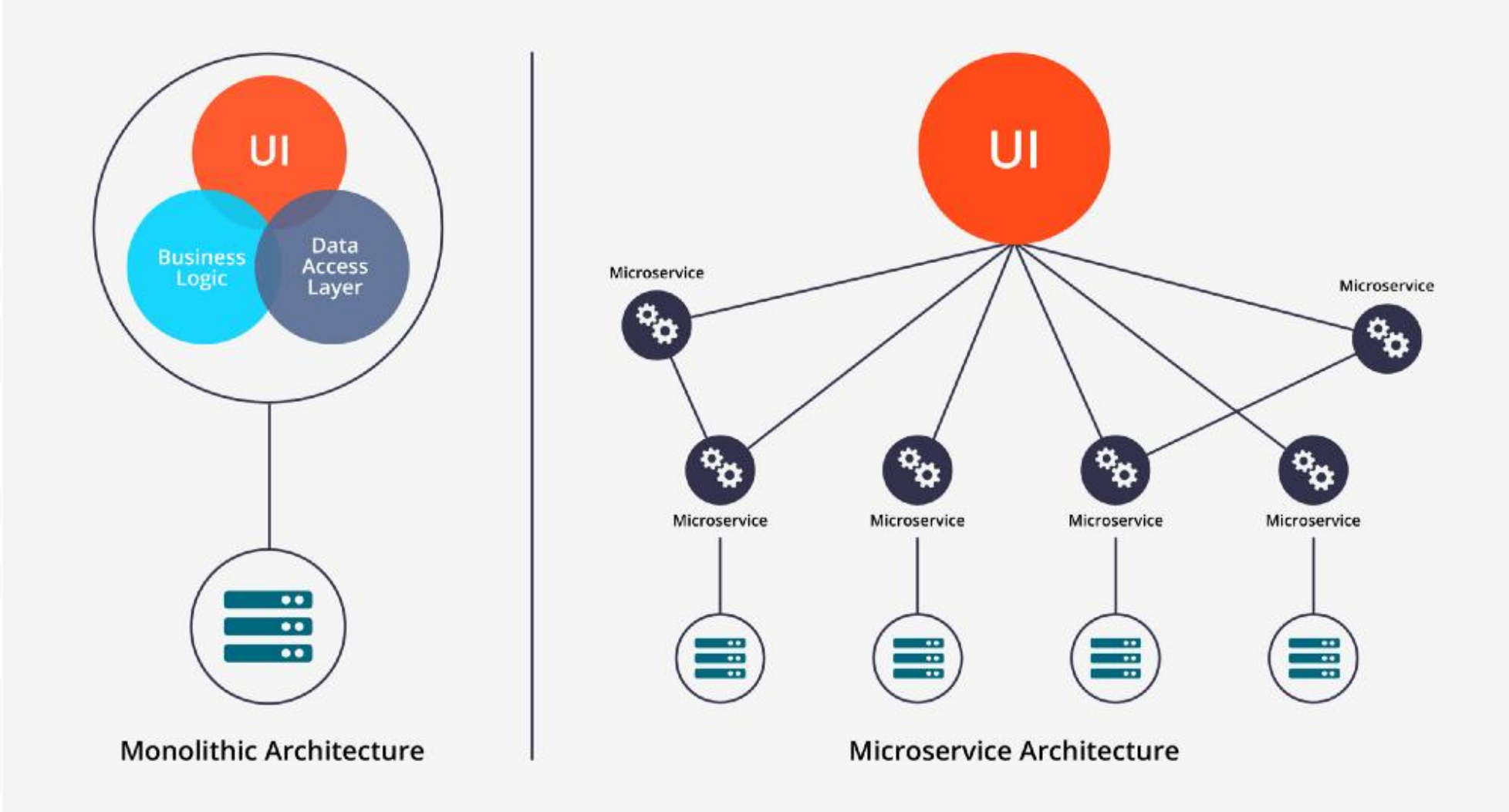
API

- se puede utilizar para conectar casi todos los procesos. A continuación, encontrarás algunos ejemplos de las funciones habituales de las API:
  - Compartir información de vuelos entre líneas aéreas y sitios de viajes
  - Usar Google Maps en una aplicación de uso compartido de transporte
  - Crear bots de chat en un servicio de mensajería
  - Integrar vídeos de YouTube en una página web
  - Automatizar flujos de trabajo entre herramientas de software B2B

Microservicios

- es una aproximación para el **desarrollo de software** que consiste en construir una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros (normalmente una API de recursos HTTP).
- Cada servicio se encarga de implementar una funcionalidad completa del negocio.
- Cada servicio es desplegado de forma independiente y puede estar programado en distintos lenguajes y usar diferentes tecnologías de almacenamiento de datos.

Microservicios



Microservicios - Características

- Los componentes son Servicios.
- Organizada en torno a las funcionalidades del negocio
- Extremos inteligentes
- Gobierno descentralizado
- Gestión de datos descentralizada
- Diseño tolerante a fallos
- Automatización de la infraestructura
- Autonomía
- Especialización

Microservicios - Ventajas

- Versátil — los microservicios permiten el uso de diferentes tecnologías y lenguajes.
- Fácil de integrar y escalar con aplicaciones de terceros.
- Los microservicios pueden desplegarse según sea necesario, por lo que funcionan bien dentro de metodologías ágiles.
- Las soluciones desarrolladas con arquitectura de microservicio permiten la mejora rápida y continua de cada funcionalidad.
- El mantenimiento es más simple y barato — con los microservicios se puede hacer mejoras de un módulo a la vez, dejando que el resto funcione normalmente.
- El desarrollador puede aprovechar las funcionalidades que ya han sido desarrolladas por terceros — no necesita reinventar la rueda, simplemente utilizar lo que ya existe y funciona.
- Un proyecto modular basado en microservicios evoluciona de forma más natural, es una forma fácil de gestionar diferentes desarrollos, utilizando los recursos disponibles, al mismo tiempo

Microservicios - Desventajas

- Debido a que los componentes están distribuidos, las pruebas globales son más complicadas.
- Es necesario controlar el número de microservicios que se gestionan, ya que cuantos más microservicios existan en una solución, más difícil será gestionarlos e integrarlos.
- Los microservicios requieren desarrolladores experimentados con un nivel muy alto de experiencia.
- Se requiere un control exhaustivo de la versión.
- La arquitectura de microservicios puede ser costosa de implementar debido a los costos de licenciamiento de aplicaciones de terceros.