**API:**

stands for Application Programming Interface. It's a set of rules, protocols, and tools that allows different software applications to communicate with each other.

APIs define the methods and data formats that developers can use to interact with a service or system, abstracting away the underlying implementation details.

simplifying the development process by providing standardized interfaces for accessing functionality.

**Communication between application and API**

Json : java script object notation

Xml : Exetensible Markup Language

The xml and json need not be compiled so generally used to write the configurations.

The browser default language is XML.

**Architectural styles : REST**

**Architectural patterns : MVC, MVP, MVVM**

**ASMX**

Active server method file

ASMX web services were designed to enable interoperable communication between different software applications over the internet.

Allow clients to invoke methods exposed by the web service and receive results in a standardized format, typically XML.

Uses SOAP + HTTP(or any other application layer protocol like FTP, HTTP, HTTPS)

typically exchange data in XML format using SOAP messages. However, they can also support other formats such as JSON by using additional serialization techniques.

Define methods by using WebMethod attribute

HTTP + REST = JSON

HTTP + SOAP = XML

**WCF**

Windows Communication Foundation

used to create a distributed and interoperable Application.

Distributed : which do not run only on single system but can run on multiple systems, which are connected over the network.

Interoperable : Application can consume or connect with another Application but it does not matter in which platform it is developed.

Defines services with help of contracts

**SOAP**

Simple object access protocol

Network protocol for exchanging data between nodes.

Uses XML format to transfer messages

built on top of HTTP which is installed in most systems.

**REST**

Representational State Transfer

Architectural style

Rest properties:

1. Rest client-server
2. Stateless
3. Layered
4. Cacheable
5. Uniformed Contract

**HATEOS : Hypertext as the engine of Application State**

component of RESTful API architecture.

With the use of HATEOAS, the client-side needs minimal knowledge about how to interact with a server.

This is made possible by the network application responding to the client’s requests with dynamically generated information through the use of hypermedia.

Now, at client side users easily interact with server by using buttons, hyper links where they generate click events.

But, in traditional API, we don’t have that functionality . At that time, the HATEOS acts as a way to address this.

It not only allows to send data, but also tells what to do next

When using HATEOS, client will be able to access the API for network application with a simple, static, restful api call.

This will enable the client to move from one application state to the next just by interacting with the details contained in the responses by the server.

**Attributes**

attributes provide a way to add metadata or declarative information to types, members, assemblies, or other program elements.

System.Attribute

The declarative info can be retrieved in run time with the help of reflection.

1. Obselete : Marks types and type methods outdated.
2. Web Method : To expose a method as an XML service
3. Serialisable : Indicates that class can be serializable

Web method : attribute, to make it available outside the class

**Docker**

Docker is a platform and set of tools that enables developers to build, deploy, and run applications using containers. Containers are lightweight, portable, and self-sufficient environments that encapsulate an application along with its dependencies, allowing it to run reliably across different environments.

Web app -> url : controller name/ action name , return type : Action result

Web API -> controller name/ => executed by http words(methods)-> http get, post, put, delete, patch , return type : string/json/xml

Query string : api/values?userId=5 & stuID=6 [in the query string, we can have the string length based on browser character support]

Content negotiation

Default value accept by browser -> xml

Default accept by postman -> json

**Content type**

Purpose: The Content-Type header is used in HTTP requests to specify the type and format of the data being sent in the request body.

Usage: When sending data in an HTTP request (e.g., submitting a form, uploading a file), the Content-Type header is included to indicate the media type of the data being transmitted.

Example: When uploading a JSON payload to a server, you might include a Content-Type header with the value application/json.

**Accept type**

Purpose: The Accept header is used in HTTP requests to specify the type of response content that the client can handle or prefers.

Usage: When making an HTTP request, the client can include an Accept header to indicate the media types it is willing to accept in the response. This allows the server to choose an appropriate representation of the resource.

Example: A client might include an Accept header with the value application/json to indicate that it prefers to receive JSON-formatted responses.

Swagger -> documentation

Calling url from browser

1. From browser can use ajax to call API
2. In angular -> HttpClient

Calling Url from c#

1. RestClient (nuget packages)

Jquery – ajax query

Services -> web service, window service

MVC app