

**PREGRADO**



UNIDAD 3 | WEB SERVICES

# **ASP.NET CORE & WEB SERVICES**

**REVIEW**

Al finalizar la semana, el estudiante comunica resultados y proceso ágil colaborativo aplicado para la implementación de componentes para las capas de interfaz y servicio, de una aplicación de lado servidor, con características innovadoras, bajo una arquitectura orientada a servicios y aplicando los principios RESTful utilizando el lenguaje C# y Microsoft .NET Framework.

---

# AGENDA

ASP.NET CORE & MVC

DEPENDENCY INJECTION

RESTFUL WEB SERVICES



# Introducción



**Tecnología**



**Framework  
de trabajo**

## Concepto

- ASP.NET Core MVC es un framework de código abierto y multiplataforma para la creación de aplicaciones modernas conectadas a Internet, como aplicaciones web y web services.
- Se diseñó para proporcionar un framework de desarrollo optimizado para las aplicaciones que se implementan tanto en la nube como en servidores dedicados en las instalaciones del cliente.

## Concepto

- ASP.NET Core MVC aprovecha el uso del patrón MVC, inyección de dependencias y una canalización de solicitudes formada por middleware.
- Es open source bajo la licencia Apache 2.0, lo que significa que está disponible de forma gratuita y la comunidad es animada para contribuir con corrección de errores y adición de nuevas características.

# Construir Web APIs y Web UI utilizando ASP.NET Core MVC

ASP.NET Core MVC proporciona features para construir web APIs y web apps:

El patrón Modelo-Vista-Controlador (MVC) ayuda a que sus API web y aplicaciones web sean comprobables.

Razor Pages es un modelo de programación basado en páginas que hace que la creación de una interfaz de usuario web sea más fácil y productiva.

El Razor markup proporciona una sintaxis productiva para las páginas de Razor y las vistas de MVC.

Los Tag Helpers permiten que el código del lado del servidor participe en la creación y representación de elementos HTML en archivos Razor.

El soporte integrado para múltiples formatos de datos y negociación de contenido permite que sus web APIs lleguen a una amplia gama de clientes, incluidos browsers y dispositivos móviles.

*Model binding* asigna automáticamente los datos de las solicitudes HTTP a los parámetros del método de acción.

*Model validation* realiza automáticamente la validación del lado del cliente y del lado del servidor.

# Program.cs

ASP.NET Core apps son creadas con templates conteniendo el código de startup de la aplicación en el archivo Program.cs, donde:

Se configuran los servicios requeridos por la app.

Se define el request handling pipeline como una serie de middleware components.

```
var builder =
WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    app.UseHosts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseAuthorization();

app.MapGet("/hi", () => "Hello!");

app.MapDefaultControllerRoute();
app.MapRazorPages();

app.Run();
```



---

# AGENDA

MVC PATTERN

ASP.NET CORE & MVC

DEPENDENCY INJECTION

RESTFUL WEB SERVICES



## Concepto

- La inyección de dependencias (DI, por sus siglas en inglés) es un patrón usado en el diseño orientado a objetos que trata de solucionar las necesidades de creación de los objetos de una manera práctica, útil, escalable y con una alta versatilidad del código.
- Tiene como finalidad solucionar el problema de mantener los componentes o capas de una aplicación lo más desacopladas posible.
- Busca que sea más sencillo reemplazar la implementación de un componente por otro.

## Concepto

- Para cumplir con dicho objetivo, DI nos permite inyectar comportamientos a componentes haciendo que nuestras piezas de software sean independientes y se comuniquen únicamente a través de una interface.

---

# AGENDA

MVC PATTERN

ASP.NET CORE & MVC

DEPENDENCY INJECTION

RESTFUL WEB SERVICES



REST

REpresentational State Transfer

# RESTful

Web Services basados en arquitectura REST.

Implementan interacción basada en métodos HTTP

Aplican URI (Uniform Resource Identifier)

Representación de información en formato JSON

# RESTful API

Application Programming Interface

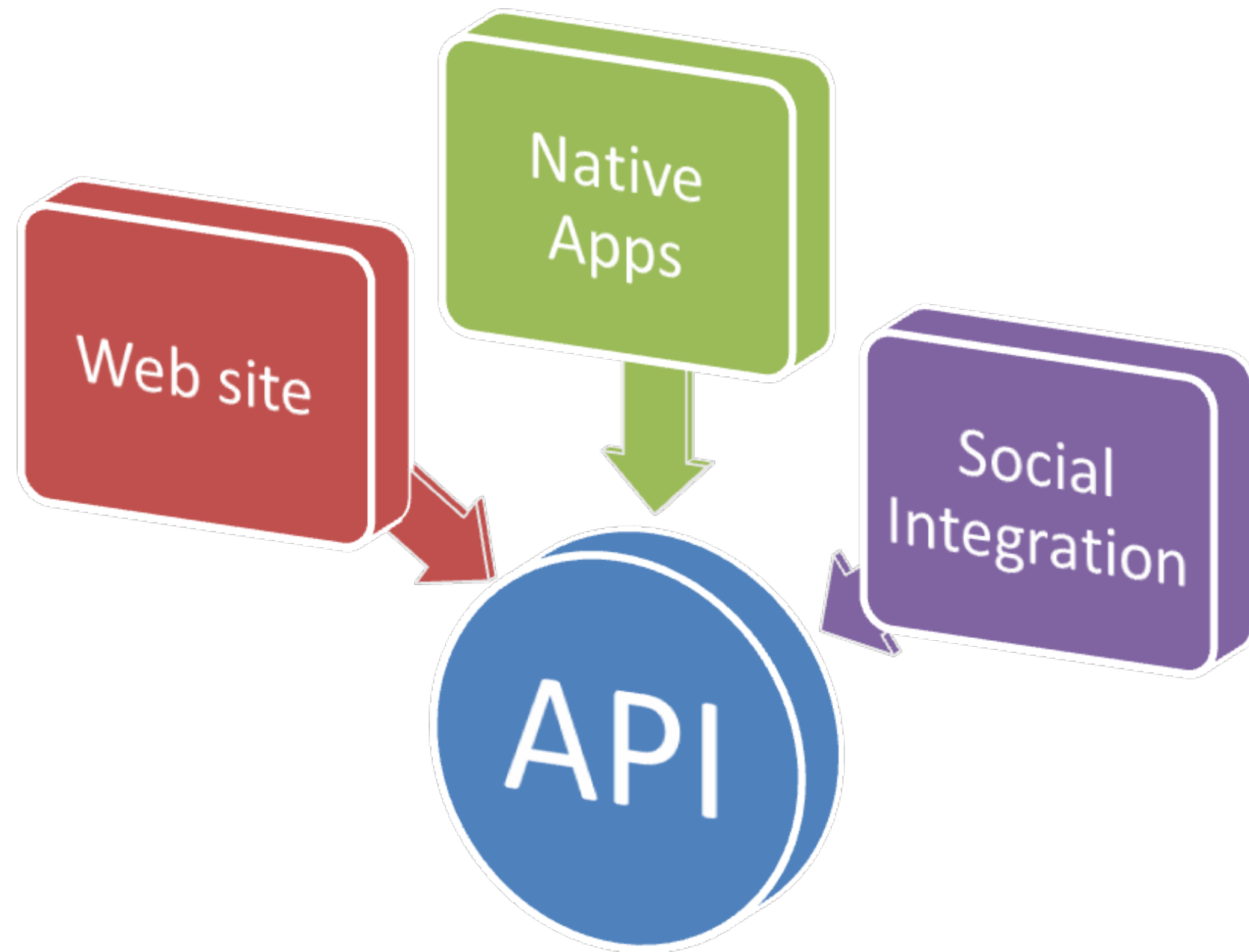
Descompone una transacción en pequeños  
módulos

Cada módulo gestiona una parte de la transacción.

Aplica métodos basados en HTTP (RFC 2616)

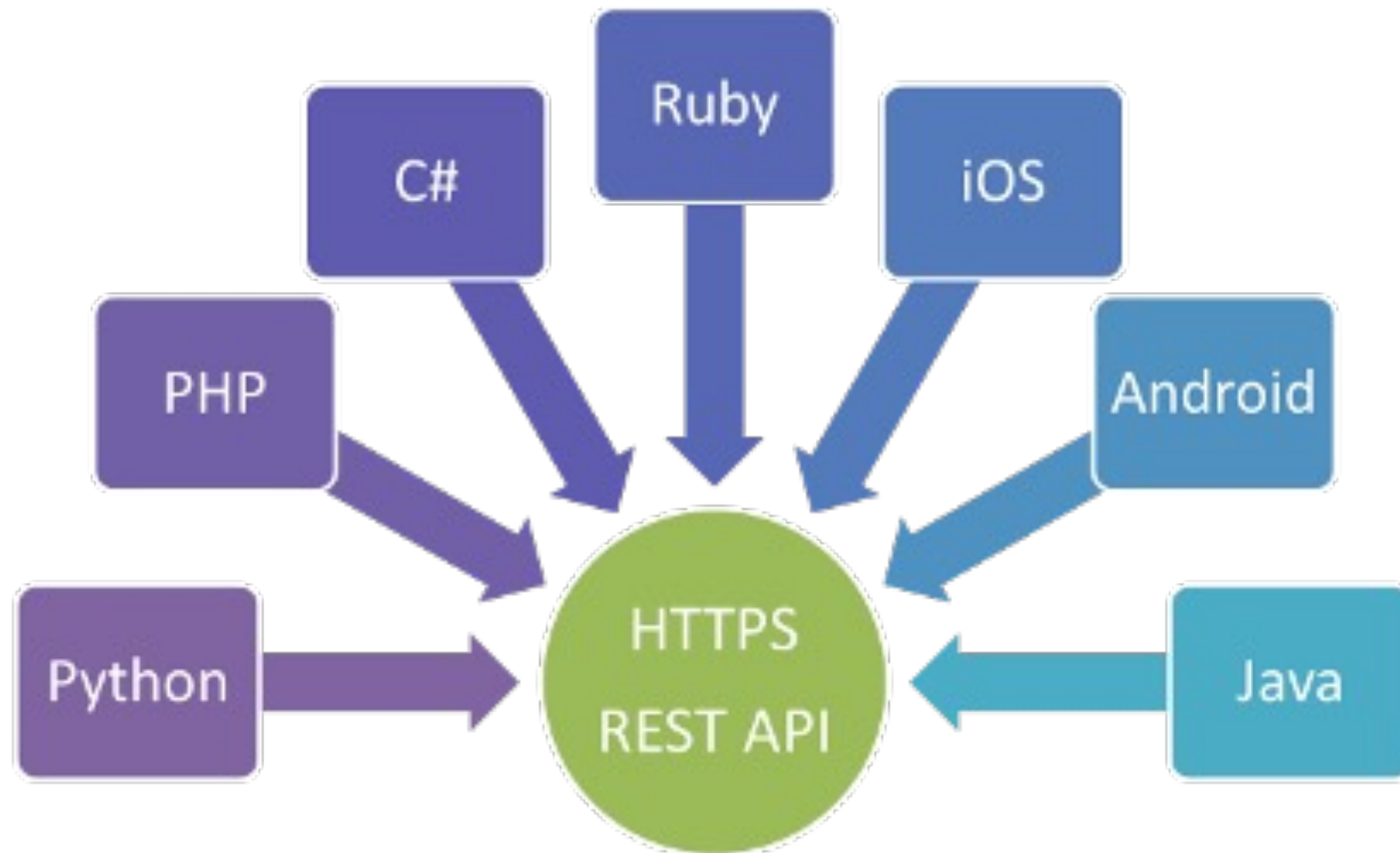


# RESTful API





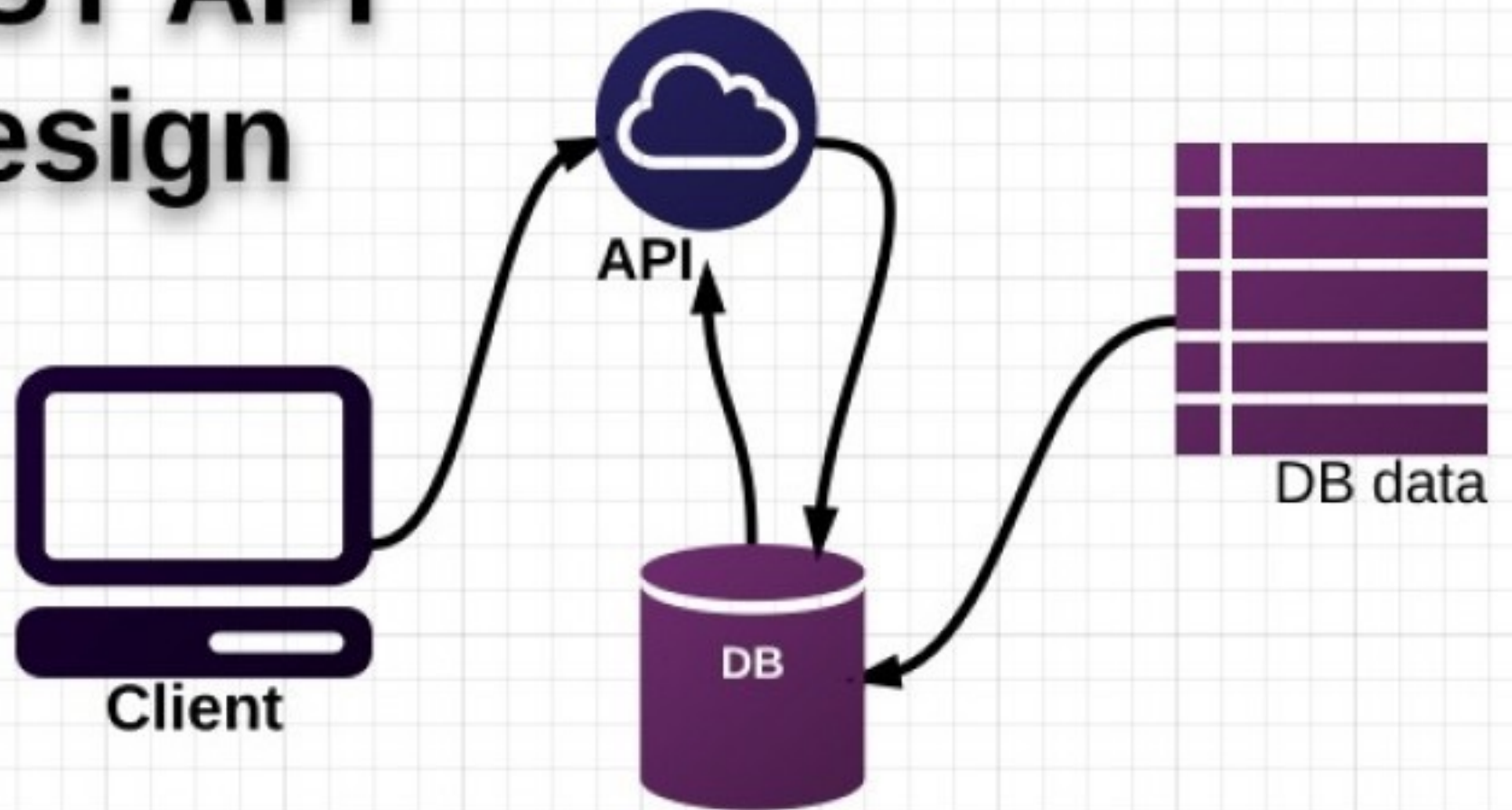
# RESTful API



# RESTful API

## REST API Design

GET	/tasks - display all tasks
POST	/tasks - create a new task
GET	/tasks/{id} - display a task by ID
PUT	/tasks/{id} - update a task by ID
DELETE	/tasks/{id} - delete a task by ID



# RESTful API

## HTTP Methods

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create To create new subordinate resource	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read To retrieve resource representation/information only	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace To update existing resource	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/Modify To make partial update on a resource	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete To delete resources	405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

# JSON

JavaScript Object Notation

# JSON

Colección de pares nombre / valor

Lista ordenada de valores

```
{ "customers": [  
    { "firstName": "John", "lastName": "Doe" },  
    { "firstName": "Anna", "lastName": "Smith" },  
    { "firstName": "Peter", "lastName": "Jones" }  
]
```

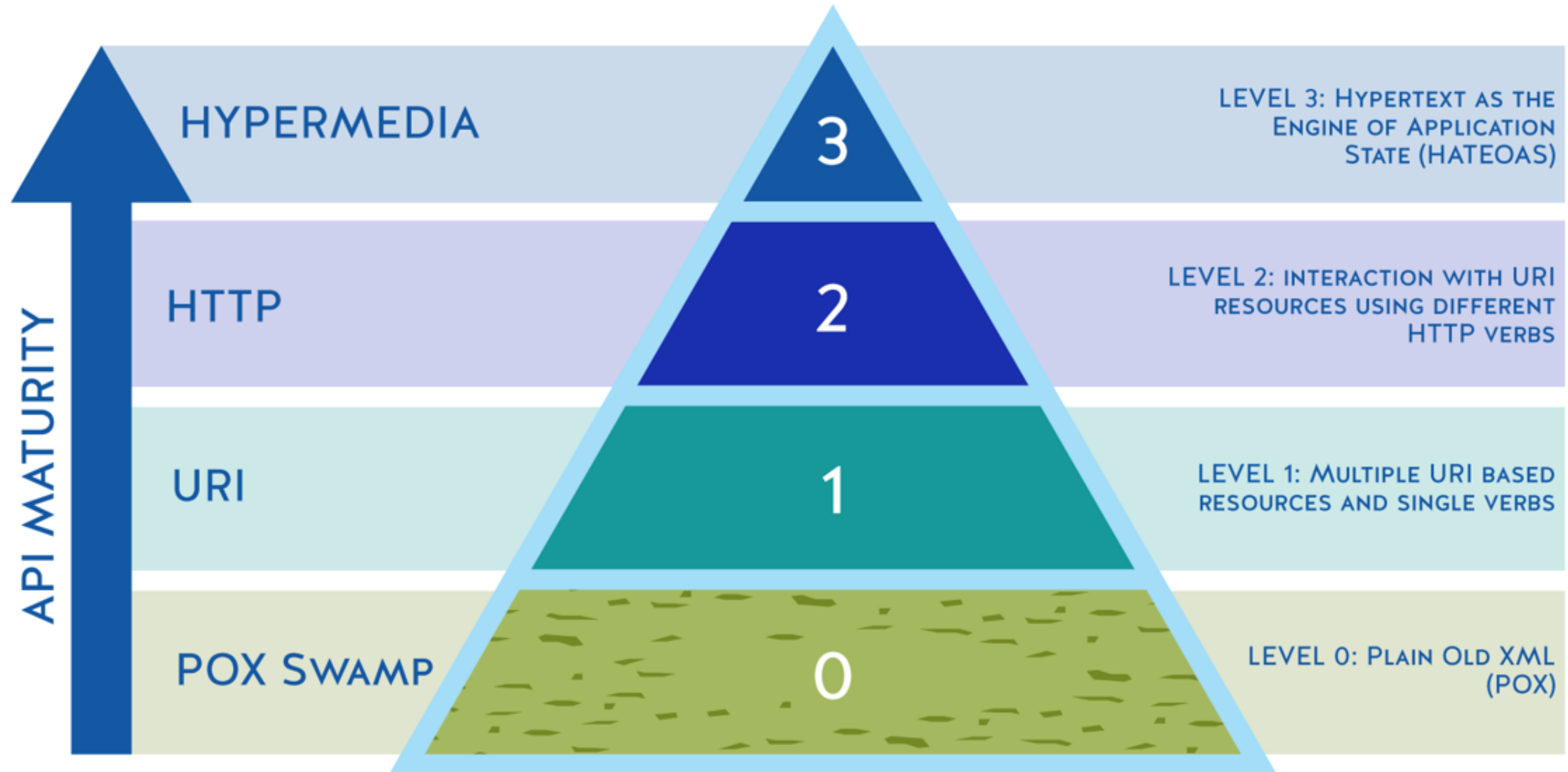
# JSON Files

Extensión .JSON

MIME Type “application/json”

# Richardson Maturity Model

## THE RICHARDSON MATURITY MODEL



# HATEOAS Driven REST APIs

HATEOAS (Hypermedia as the Engine of Application State)

Es una restricción de la arquitectura de la aplicación.

*Hypermedia* se refiere que contiene enlaces a otras formas de media como images, movies, text.

Por ejemplo: HTTP GET `http://api.domain.com/management/departments/10`

```
{
  "departmentId": 10,
  "departmentName": "Administration",
  "locationId": 1700,
  "managerId": 200,
  "links": [
    {
      "href": "10/employees",
      "rel": "employees",
      "type": "GET"
    }
  ]
}
```

<https://restfulapi.net/hateoas/>



# REST Resource Naming

REST Resource Naming Guide

<https://restfulapi.net/resource-naming/>

# REST API Versioning

## URI Versioning

<https://example.com/api/v1>

<https://api.example.com/v1>

<https://apiv1.example.com>

## Otros tipos

<https://restfulapi.net/versioning/>

---

# RESUMEN

## Recordemos

El patrón de desarrollo MVC, el cual nos permite descomponer nuestra aplicación en capas o componentes que facilitan el manejo del proyecto.

En la actualidad las aplicaciones web requieren ser seguras, flexibles y escalables, es por eso que Microsoft lanzó una nueva tecnología llamada .NET Core. Esta tecnología se complementa con el uso del framework ASP .NET Core.



---

# RESUMEN

## Recordemos

Una alternativa para manejar eficientemente la creación de objetos en nuestro proyecto es utilizar el patrón de inyección de dependencias.

RESTful viene de REpresentational State Transfer

A nivel de representación de información de web services, uno de las principales combinaciones es RESTful API + JSON



---

# REFERENCIAS

## Para profundizar

<https://www.neosoft.es/blog/que-es-una-aplicacion-web/>

<https://si.ua.es/es/documentacion/asp-net-mvc-2/modelo-vista-controlador.html>

<https://msdn.microsoft.com/es-es/magazine/mt694084.aspx>

<https://github.com/jahbenjah/little-aspnetcore-book/blob/spanish/ElPeque%C3%B1oLibroDeASPNETCore.pdf>

[https://www.youtube.com/watch?v=Gj\\_EbIN4P5w](https://www.youtube.com/watch?v=Gj_EbIN4P5w)



# PREGRADO

## Ingeniería de Software

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería



### UPC

Universidad Peruana  
de Ciencias Aplicadas

Prolongación Primavera 2390,  
Monterrico, Santiago de Surco  
Lima 33 - Perú  
T 511 313 3333  
<https://www.upc.edu.pe>

***exígete, innova***