

Semana 1

**Curso:
Desarrollo de ambiente
web**

Ciclo: 2023-2

Introducción



DESARROLLO DE APLICACIONES WEB

INTRODUCCIÓN

CONTENIDO:

- Presentación.**
- Revisión del sílabo.**
- Calificación del curso.**
- Proyecto web.**
- Compromiso del alumno.**



Desarrollo de ambiente web – Semana 1

INTRODUCCIÓN



Presentación

- Del Docente:
 - Presentación personal.
- Del Alumno:
 - Nombre completo
 - Edad
 - Experiencia en desarrollo web
 - Expectativas del curso



Desarrollo de ambiente web – Semana 1

REVISIÓN DEL SÍLABO



Desarrollo de ambiente web – Semana 1

CALIFICACIÓN DEL CURSO



Calificación del curso

Promedio de
Evaluación
Permanente
(PEP)

PROMEDIO DE EVALUACIÓN PERMANENTE (PEP) 60%		
Tipo de evaluación	Descripción	Ponderación %
Prácticas calificadas	Cinco (5) prácticas calificadas (ninguna se anula)	35%
Proyecto Web Empresarial	Proyecto grupal desarrollado con Visual Studio	60%
Participación en clase	Participación en clase, actividades en aula.	5%

Promedio Final
(PF)

$$\text{PF} = (0,20 \times \text{EP}) + (0,60 \times \text{PEP}) + (0,20 \times \text{EF})$$

EP: Examen Parcial
EF: Examen Final

Desarrollo de ambiente web – Semana 1

PROYECTO WEB



Proyecto web

- Se trabajará con tres (3) grupos.
- Enviar nombre del grupo e integrantes (semana 01).
- El tema y requisitos del proyecto se asignará a los grupos en la semana 05 por el curso de Ing. De Software.
- Entrega Trabajo Parcial backend API Web (semana 09).
- Entrega Trabajo Final frontend y backend (semana 15).

Desarrollo de ambiente web – Semana 1

COMPROMISO DEL ALUMNO



Compromiso del alumno

- Respeto.
- Preguntar.
- Investigar.



Semana 1

**Curso:
Desarrollo de ambiente
web**

Ciclo: 2023-2

**Unidad I: Introducción a las
aplicaciones web**



UNIDAD DE APRENDIZAJE I: INTRODUCCIÓN A LAS APLICACIONES WEB

RESULTADOS DE APRENDIZAJE:

- Comprende el ecosistema .NET.
- Comprende sobre la arquitectura de aplicaciones web estáticas y dinámicas.
- Entiende la diferencia sobre .NET Framework y .NET Core.
- Comprende el patrón de diseño MVC.
- Entiende la importancia de la programación orientada a objetos y el principio SOLID para la construcción de aplicaciones web empresariales.
- Comprende la comunicación de información entre los diversos componentes web.

CONTENIDO:

- 1.1. Introducción a .NET.
- 1.2. Introducción al desarrollo de aplicaciones web.
- 1.3. Páginas estáticas y dinámicas.
- 1.4. Fundamentos de HTML y CSS.
- 1.5. ASP.NET Framework.
- 1.6. ASP.NET Core.

Desarrollo de ambiente web – Semana 1

INTRODUCCIÓN A .NET



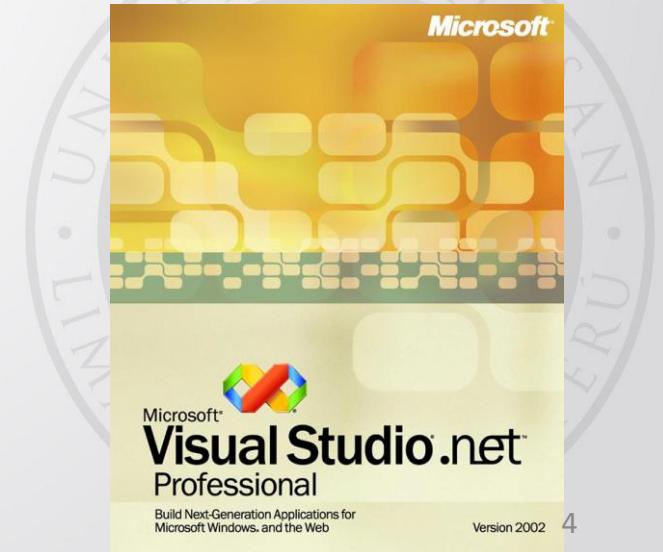
Introducción a .NET

- Acontecimientos

- Anders Hejlsberg: Padre del C# (antes J++) y colaborador con Turbo Pascal, Delphi.
- Necesidad de mejorar Visual Studio 6 (1998).
- Respuesta de Microsoft sobre Sun Microsystems.
- Lanzamiento Febrero 2002: Visual Studio .NET.
- .NET Framework 1.0



Anders Hejlsberg



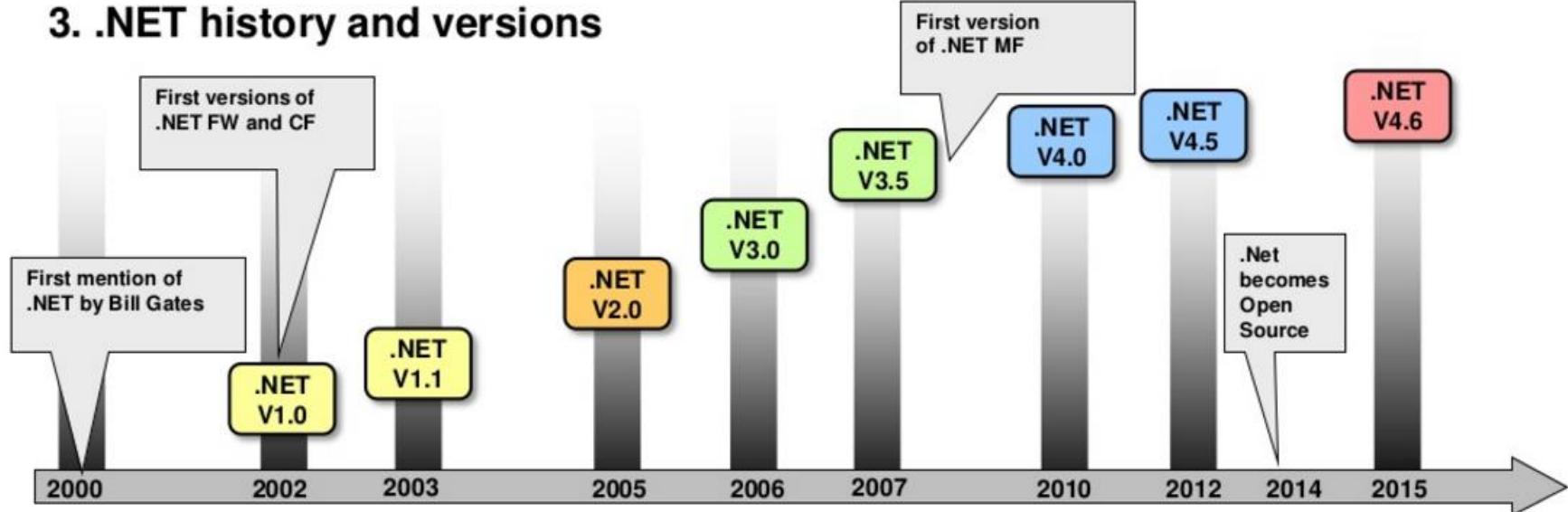
.NET Mitos y Verdades

- ¿Qué **NO** es .NET?
 - Un sistema operativo.
 - Un lenguaje de programación.
 - Un entorno de desarrollo(IDE's)
- ¿Qué **SI** es .NET?
 - Un entorno de ejecución administrado.
 - Un conjunto de lenguajes de programación.
 - Un conjunto de bibliotecas y clases reutilizables.

“.NET es un estándar de ECMA que tiene distintas implementaciones: .NET Framework, Mono, Unity y, ahora, .NET Core.”

.NET Framework

3. .NET history and versions



.NET Framework:

Standard framework for desktop and server hosts.

.NET Compact Framework:

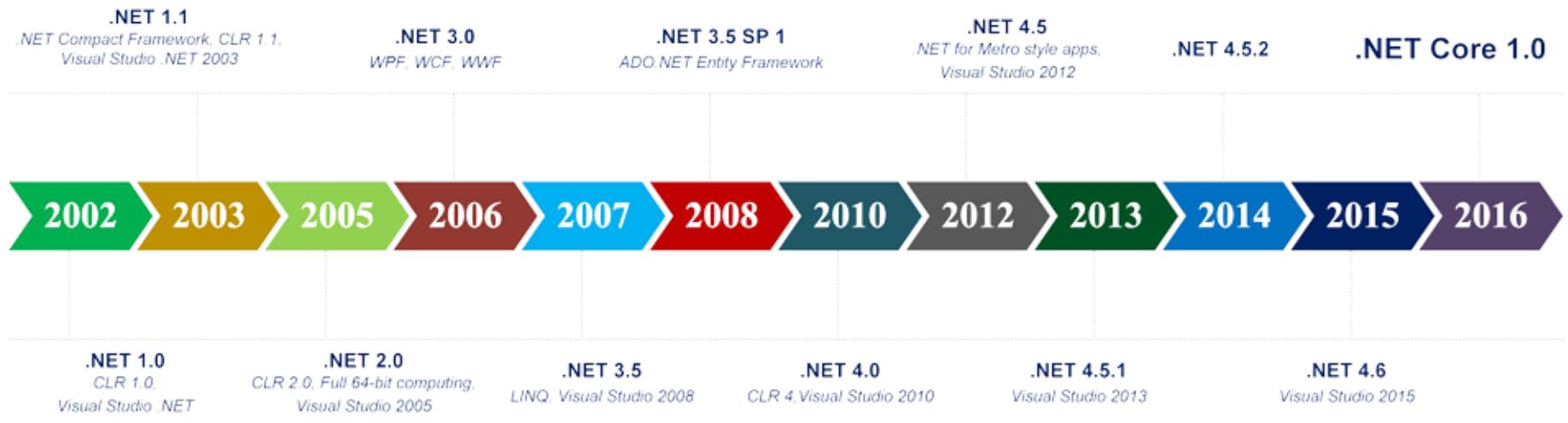
Stripped-down .NET FW for embedded devices, based on Windows CE / Windows Embedded Compact.

.NET Micro Framework:

Variant for „deeply“ embedded devices (no OS required, .NET is the OS).

Timeline .NET

The release history of .NET Framework



.NET
Core

2022

.NET 6

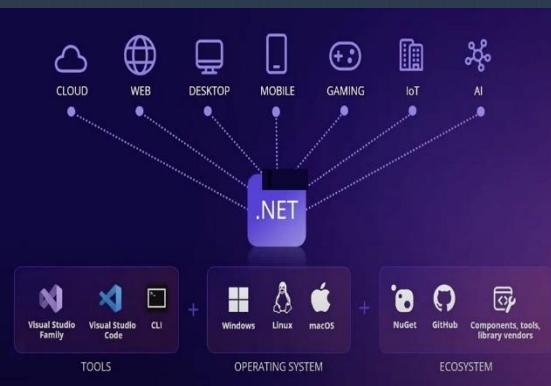
.NET
Framework

.NET Framework 4.8

Supported versions

Version	Status	Latest release
.NET 6.0	Preview ⓘ	6.0.0-preview.2
.NET 5.0 (recommended)	Current ⓘ	5.0.4
.NET Core 3.1	LTS ⓘ	3.1.13
.NET Core 2.1	LTS ⓘ	2.1.26

Download
.NET 7



.NET Framework & .NET Core

- .NET Framework: Es un framework para desarrollar aplicaciones para Windows (desktop, mobile, services), ***Nivel del madurez: Alta.***
- .NET Core: Entorno de ejecución Cross-Platform. ***Nivel del madurez: media-alta.***

¿Cuando utilizar .NET Core?

- .NET Framework: Es un framework para desarrollar aplicaciones para Windows (desktop, mobile, services), **Nivel de madurez: Alta.**
- .NET Core: Entorno de ejecución Cross-Platform. Nivel de madurez: **Media-Alta.**

¿Cuando utilizar .NET Core?

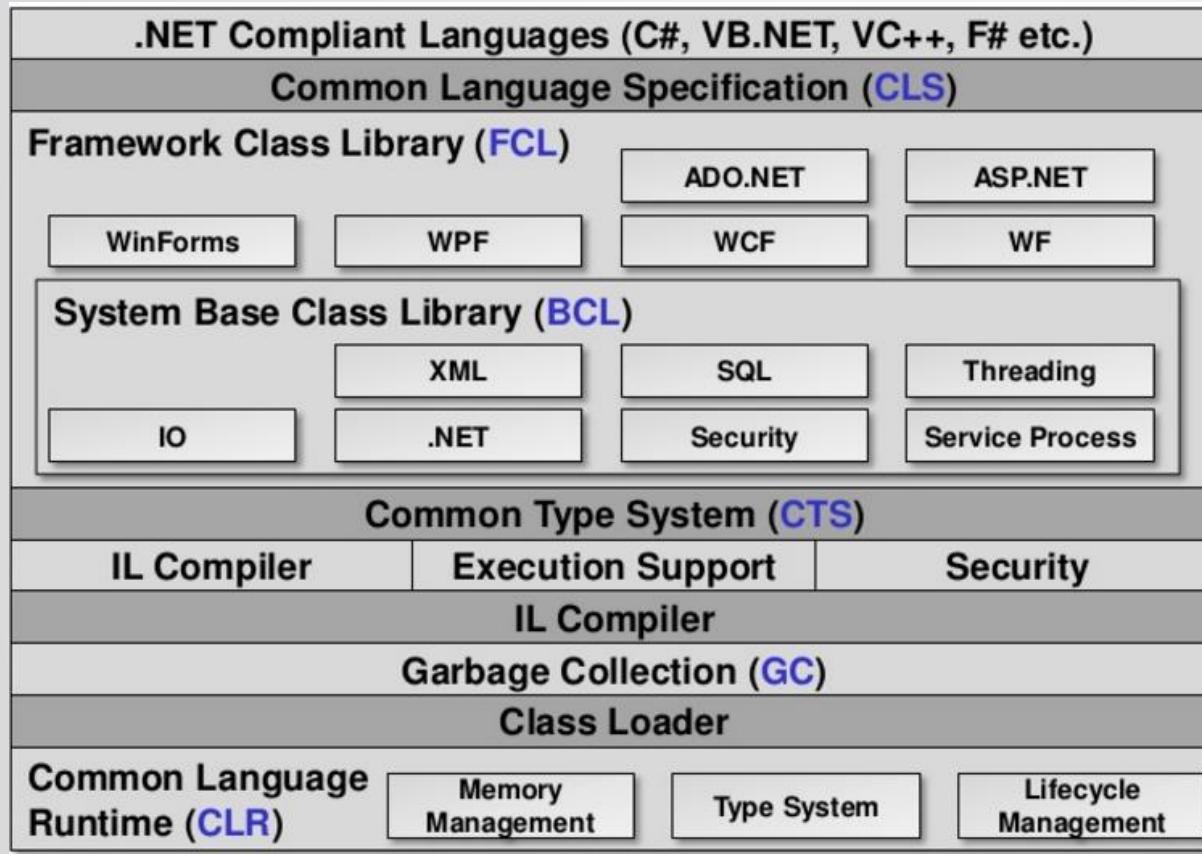
Recomendable Utilizar .NET CORE cuando:

- Nuevos proyectos.
- Cross-Platform.
- Arquitectura basada en microservicios.
- Escalable y con altas exigencias de rendimiento.

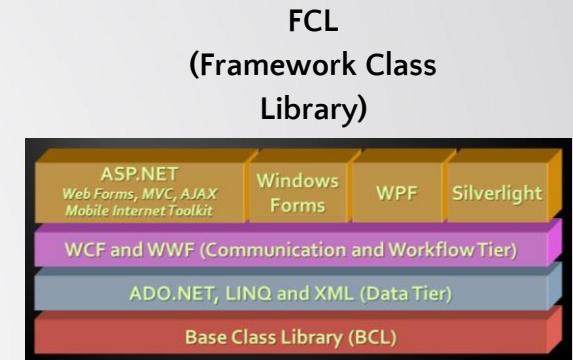
NO recomendable Utilizar .NET CORE cuando

- Migración de aplicaciones actuales.
- Dependencia de Entity Framework, Windows Communication Foundation, Workflow Foundation.

.NET Framework Overview

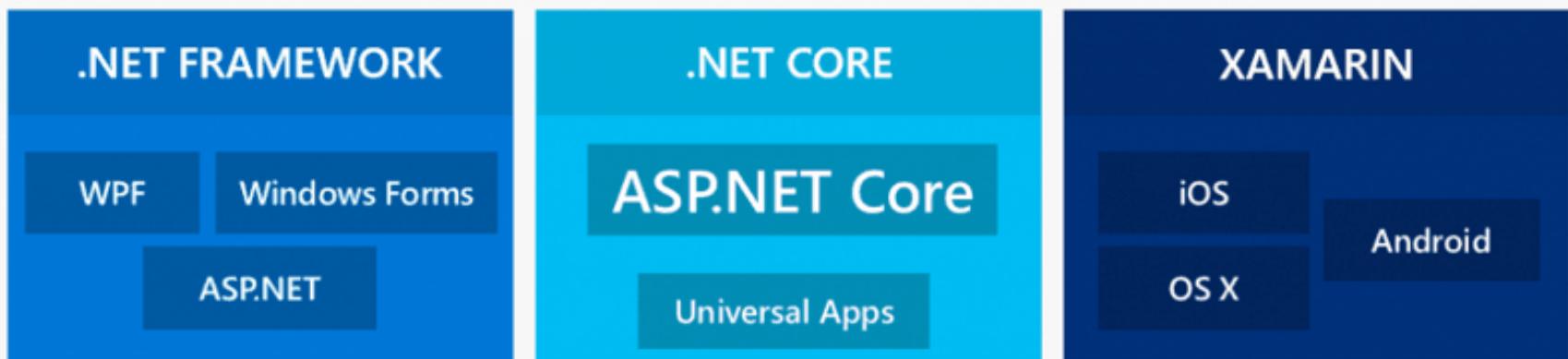


Fuente: Peter Egli 2015



CLR (Common Language Runtime)

.NET Framework Overview



.NET STANDARD LIBRARY
One library to rule them all

COMMON INFRASTRUCTURE

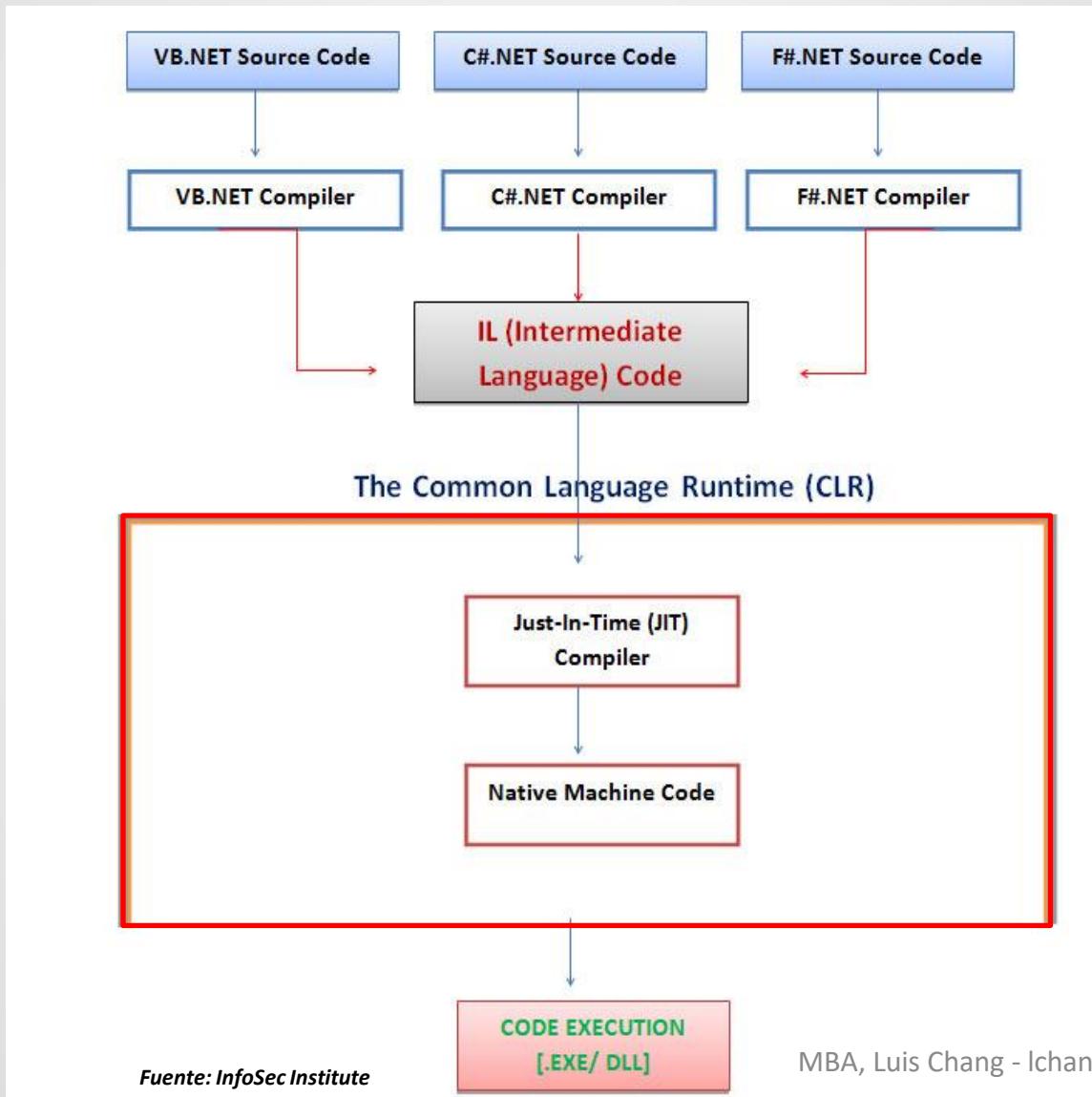
Compilers

Languages

Runtime components

Command Line

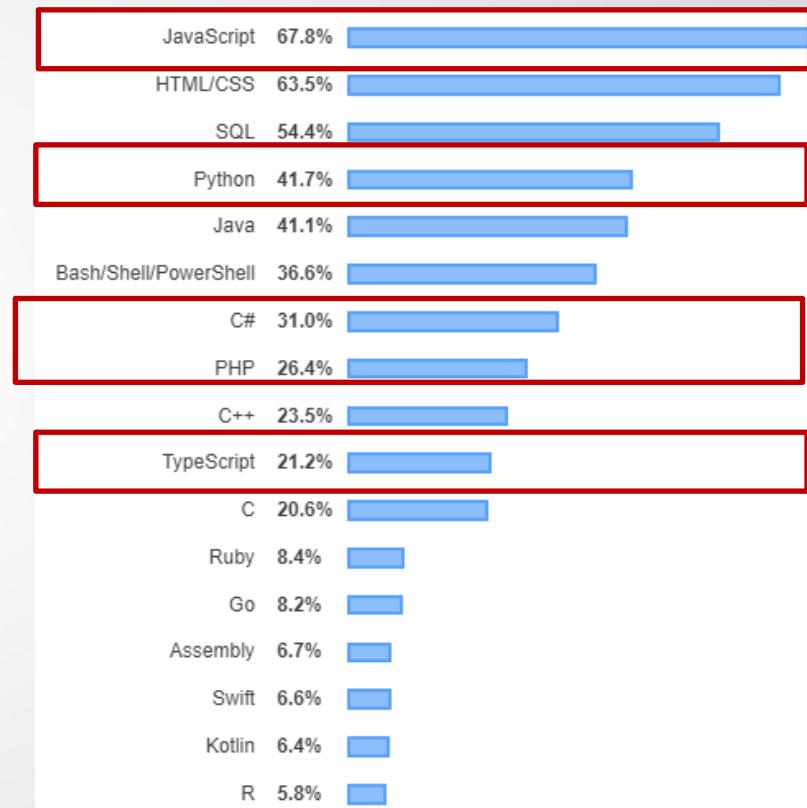
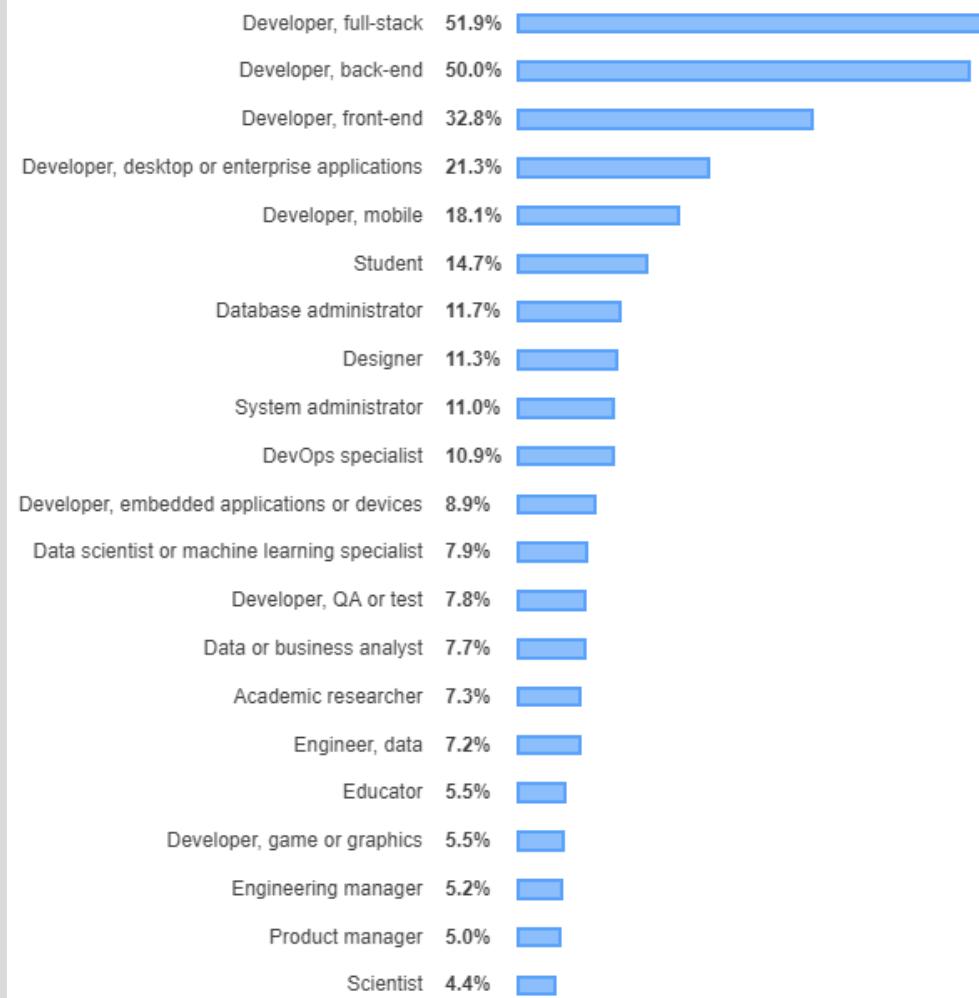
Common Language Runtime (CLR)



Desarrollo de ambiente web – Semana 1

INTRODUCCIÓN AL DESARROLLO DE APLICACIONES WEB

Tendencias sobre el desarrollo de software

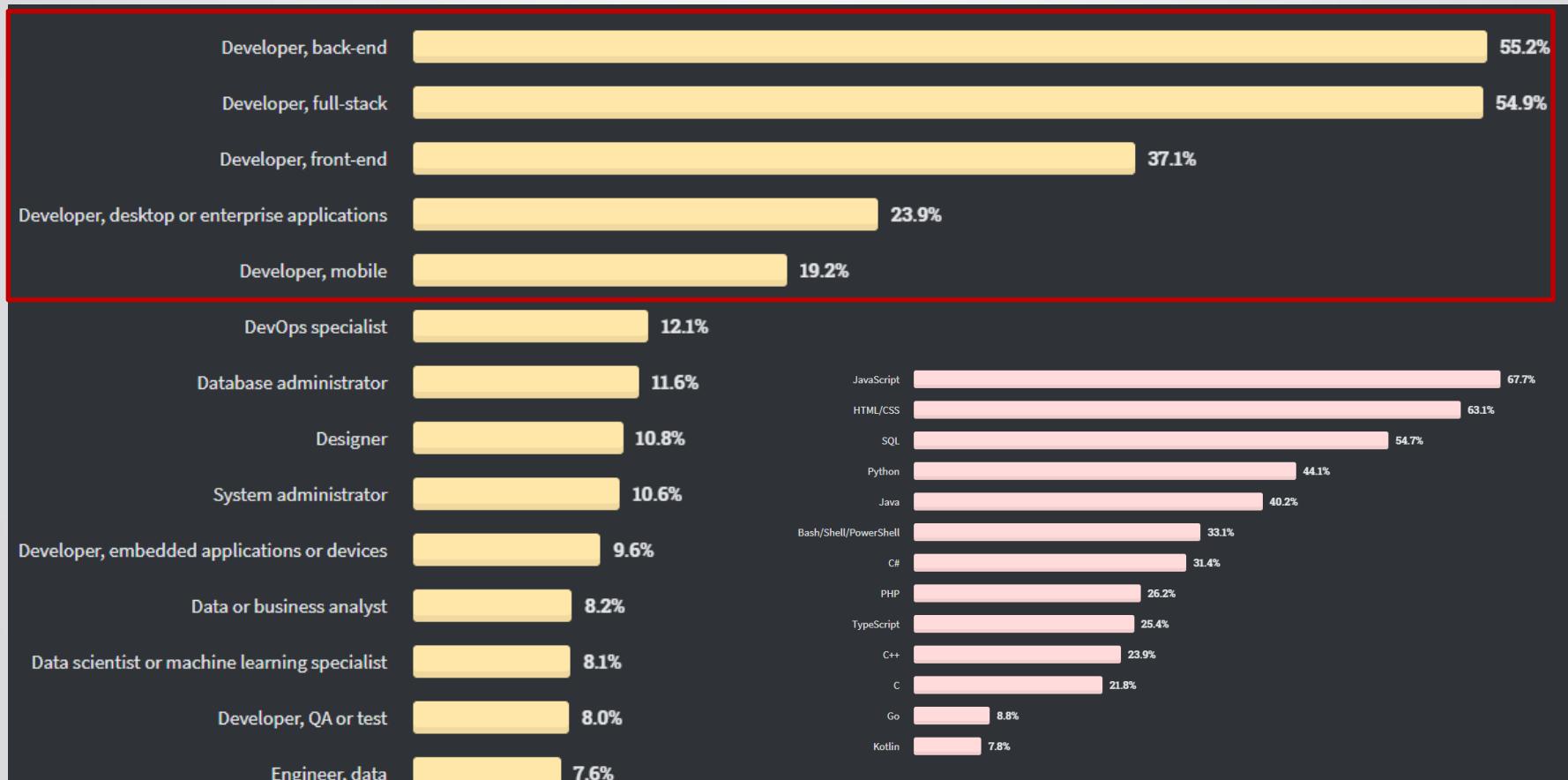


Fuente: [Stack Overflow Developer Survey Results 2019](#)

90,000 responses;

15

Tendencias sobre el desarrollo de software

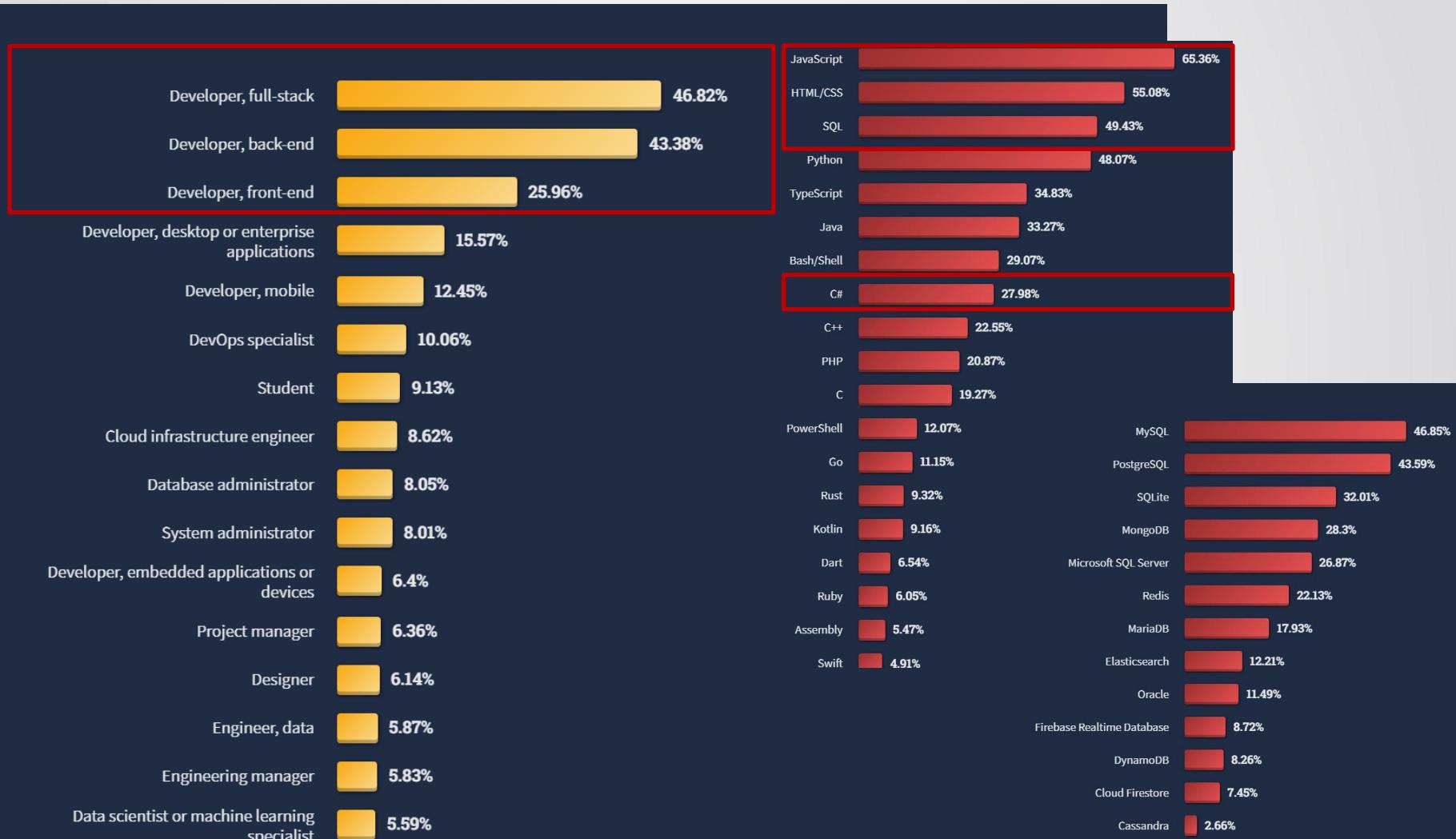


Fuente: [Stack Overflow Developer Survey Results 2020](#)

90,000 responses;

16

Tendencias sobre el desarrollo de software



Fuente: [Stack Overflow Developer Survey Results 2022](#)

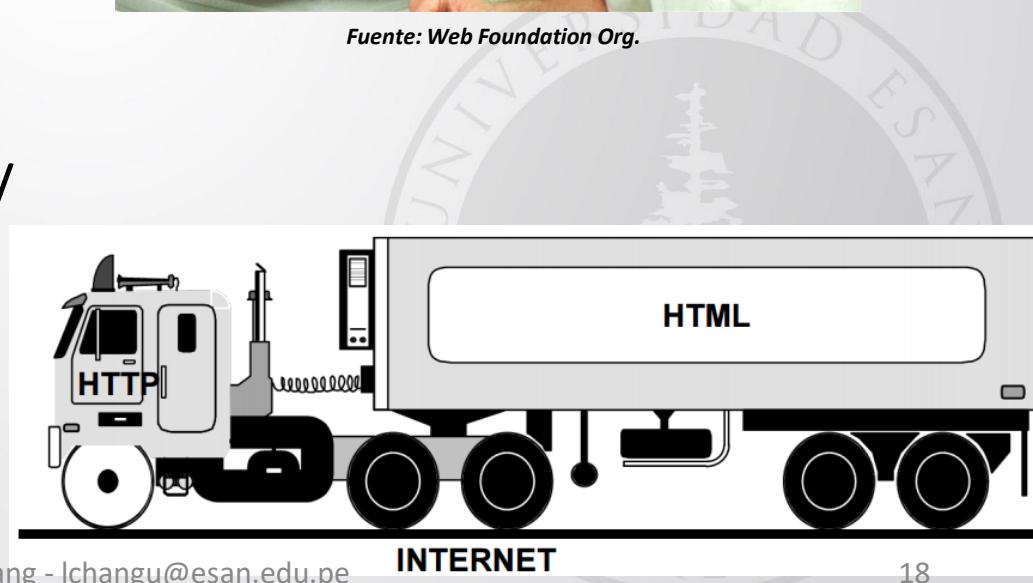
61,302 responses;

World Wide Web - History

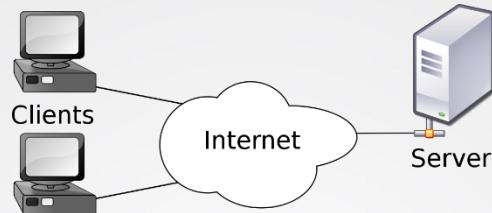
- Creada en 1989 por Tim Berners.
- Basado en hipertextos.
- Hyper Text Markup Language.
- Hyper Text Transfer Protocol.
- Uniform Resource Locator/Identifier



Fuente: Web Foundation Org.



Web Page – Request/Response



```
<html>
  <head>
    <meta content="text/html;
      charset=UTF-8" http-equiv="Content-Type"/>
    <title>My lovely web page </title>
  </head>
  <body>
    <h1>This is my lovely web page</h1>
    <p>
      It has lots of lovely content. It has some
      <em>emphasized text</em>
      and look at this, a blockquote:
    </p>
    <blockquote>
      <p>You fools, I will destroy you
        all!</p>
    </blockquote>
    <h2>And here's a subheading</h2>
    <p>That about covers it, I think</p>
    <hr/>
  </body>
</html>
```

This is my lovely web page

It has lots of lovely content. It has some *emphasized text* and look at this, a blockquote:

You fools, I will destroy you all!

And here's a subheading

That about covers it, I think

Desarrollo de ambiente web – Semana 1

PÁGINAS WEB ESTÁTICAS Y DINÁMICAS



Estándares Web

- Componentes básicos: **HTML, CSS y JavaScript**.
- **HTML**: Hyper Text Markup Language, es un lenguaje de **ETIQUETAS** utilizado en la capa de presentación de las páginas web.
- **CSS**: Cascading Style Sheets, es un lenguaje para definir el “estilo” de la capa de presentación. Describe como debe ser renderizado el elemento estructurado en pantalla.
- **JavaScript**: Es un lenguaje de programación multiplataforma y orientado a objetos, no requiere compilación, funciona del lado del cliente y JS le permite agregar interactividad y características dinámicas a su sitio web, pero tiene limitaciones, lo que nos lleva a los lenguajes de programación del servidor y a las páginas web dinámicas.



Páginas web estáticas

- Diseñadas para mostrar información permanente.
- De uso informativo.
- No se utilizan Base de Datos.
- Interactúan por medio hipervínculos o enlaces.
- Ventajas económicas.
- Fácil implementación.
- Edición manual.
- Enfocados a proyectos pequeños.



Páginas web dinámicas



Desarrollo de ambiente web – Semana 2

FUNDAMENTOS DE HTML Y CSS

Recursos necesarios



https://www.w3schools.com/html/html_exercises.asp

<https://learngitbranching.js.org/>

https://sqlzoo.net/wiki/SQL_Tutorial

Desarrollo de ambiente web – Semana 1

ASP.NET FRAMEWORK



.NET FRAMEWORK

.NET Compliant Languages (C#, VB.NET, VC++, F# etc.)

Common Language Specification (**CLS**)

Framework Class Library (**FCL**)

WinForms

WPF

ADO.NET

ASP.NET

WCF

WF

System Base Class Library (**BCL**)

IO

XML

SQL

Threading

.NET

Security

Service Process

Common Type System (**CTS**)

IL Compiler

Execution Support

Security

IL Compiler

Garbage Collection (**GC**)

Class Loader

Common Language
Runtime (**CLR**)

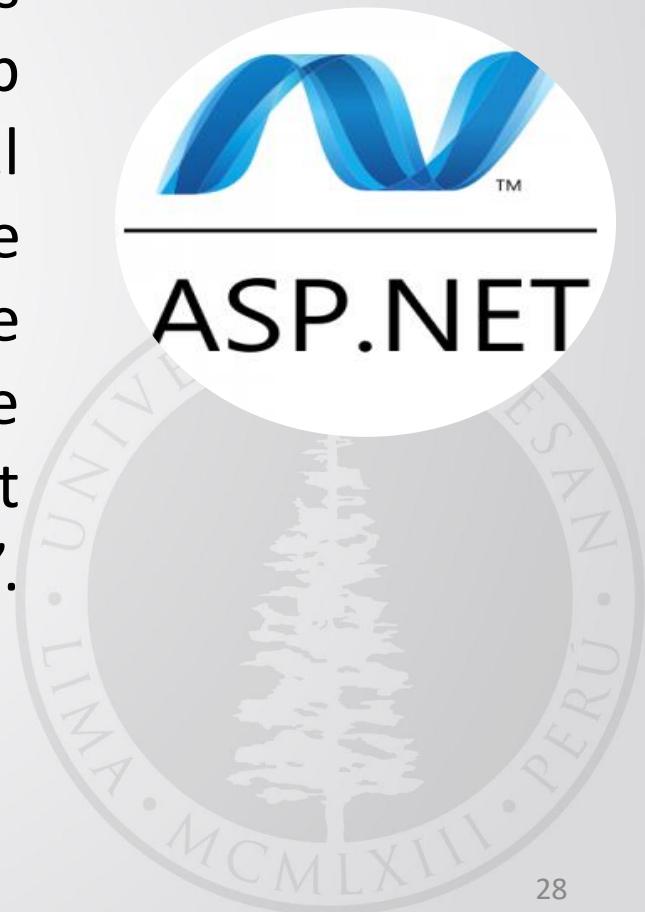
Memory
Management
MBA, Jinsong Chang jchangu@esan.edu.pe

Type System

Lifecycle
Management

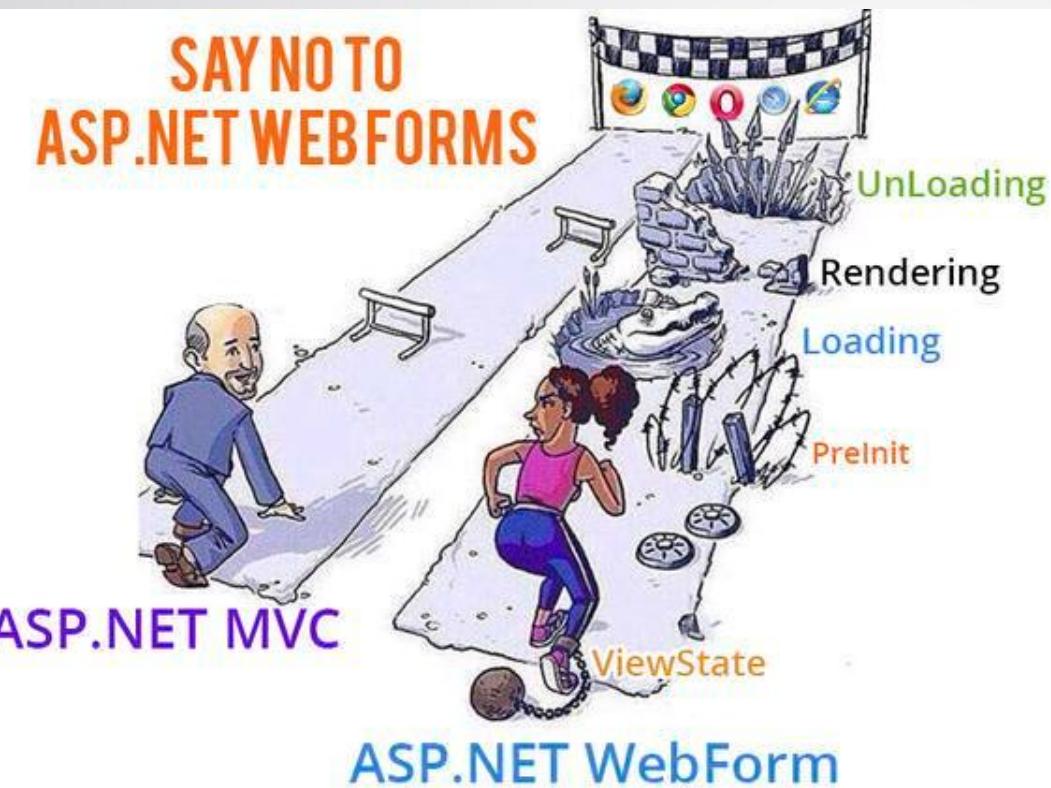
Active Server Page .NET

“Es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web empresariales con el código mínimo. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic, C#, JScript .NET y J#”.

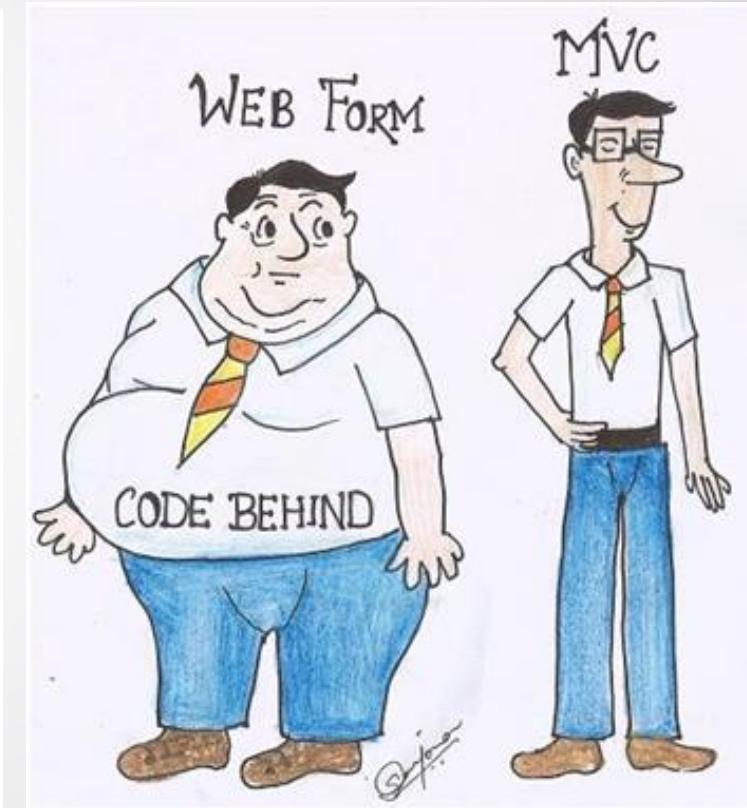


Fuente: MSDN – ASP.NET

ASP.NET: Web Forms & MVC



Fuente: Kemal Durand



Fuente: CodeProject: Web Forms vs MVC

Desarrollo de ambiente web – Semana 1

ASP.NET CORE

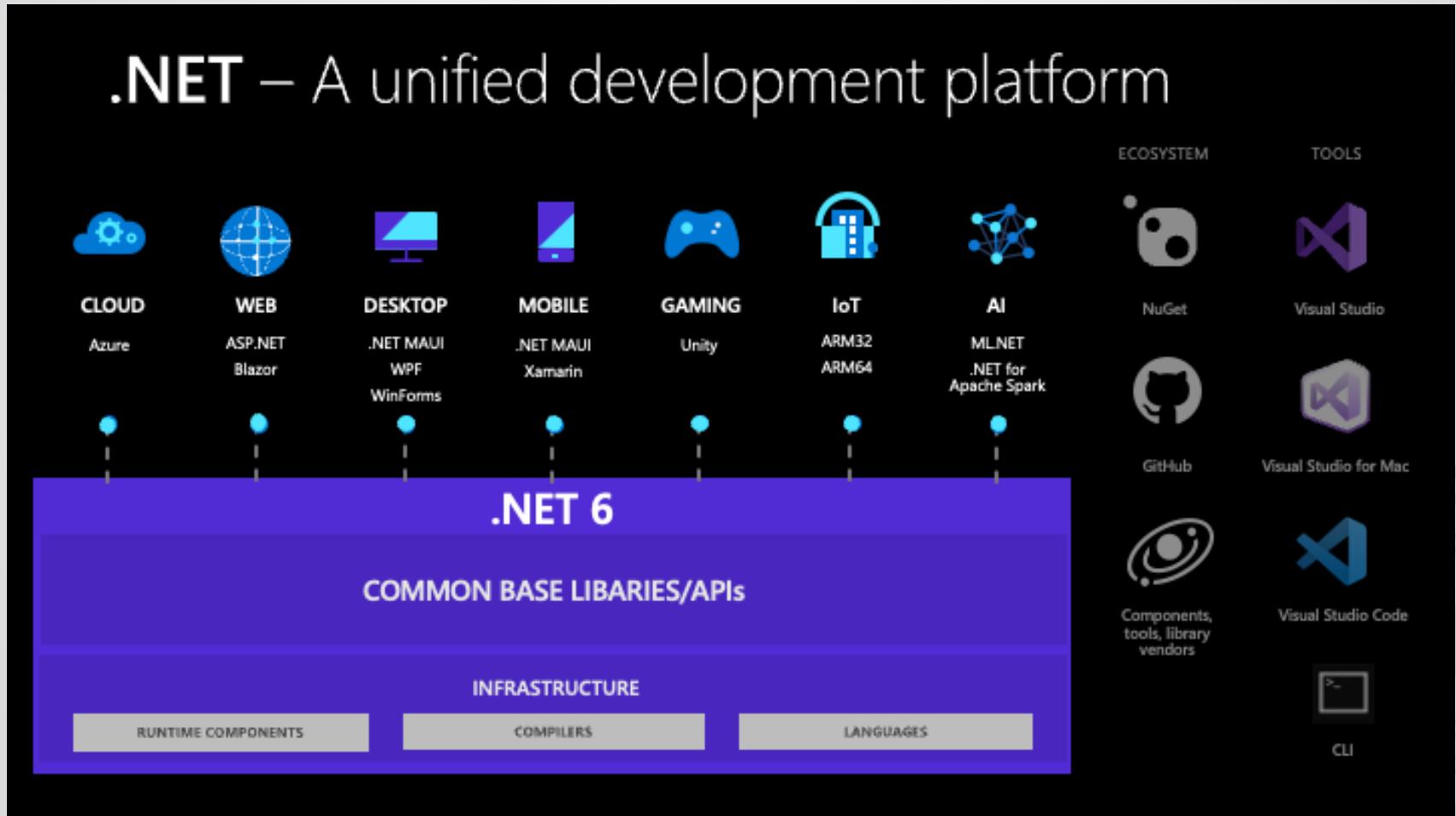


Overview .NET Core



Overview .NET Core

.NET – A unified development platform



Referencias

- Price,M. (2020). C# 9 and .NET 5 - Modern Cross-Platform Development. Birmingham: Packt Publishing Ltd. Chapter 1 Pages 8 – 17
- Frain, B. (2020). Responsive Web Design with HTML5 and CSS. Birmingham: Packt Publishing Ltd. Chapter 2 Pages 25 - 53



Semana 3

**Curso:
Desarrollo de ambiente
web**

Ciclo: 2023-2

**Unidad I: Introducción a las
aplicaciones web**



UNIDAD DE APRENDIZAJE I: INTRODUCCIÓN A LAS APLICACIONES WEB

RESULTADOS DE APRENDIZAJE:

- Comprende el ecosistema .NET.
- Comprende sobre la arquitectura de aplicaciones web estáticas y dinámicas.
- Entiende la diferencia sobre .NET Framework y .NET Core.
- Comprende el patrón de diseño MVC.
- Entiende la importancia de la programación orientada a objetos y el principio SOLID para la construcción de aplicaciones web empresariales.
- Comprende la comunicación de información entre los diversos componentes web.

CONTENIDO:

- 1.10. REST API.
- 1.11. HTTP Headers.
- 1.12. HTTP Request Methods.
- 1.13. Http Status Code.
- 1.14. Access Control Http.
- 1.15. Postman.
- 1.16. Pruebas de endpoints con Postman.

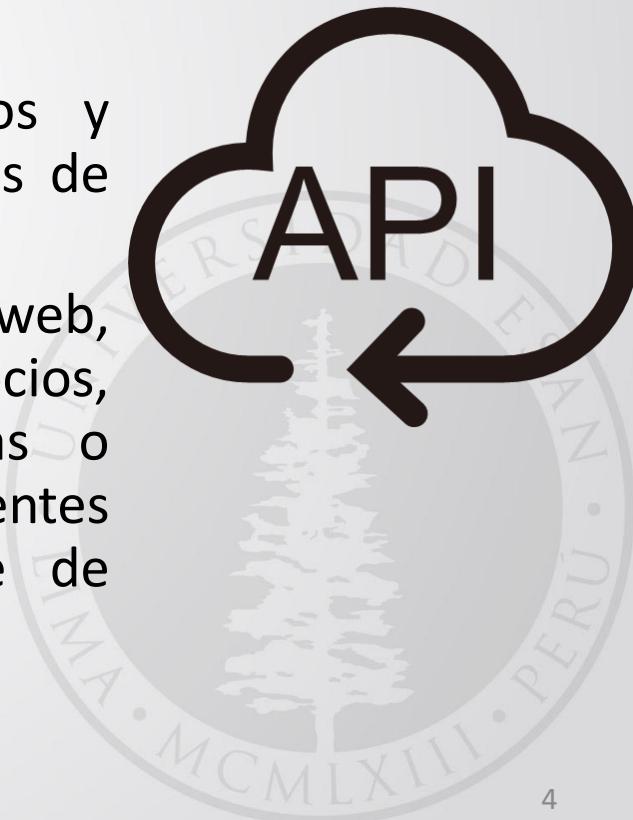
Desarrollo de ambiente web – Semana 3

REST API



API

- **Application Programming Interface**
 - Es un conjunto de productos y servicios tecnológicos que permiten comunicarse a través de internet.
 - Es un conjunto de reglas, protocolos y herramientas para construir aplicaciones de software.
 - Permite integrar diferentes aplicaciones web, móviles, automatizar procesos de negocios, extraer datos de diferentes sistemas o construir integraciones entre diferentes aplicaciones sin importar el lenguaje de programación utilizado.





BACK-END

FRONT-END

APIs

REST

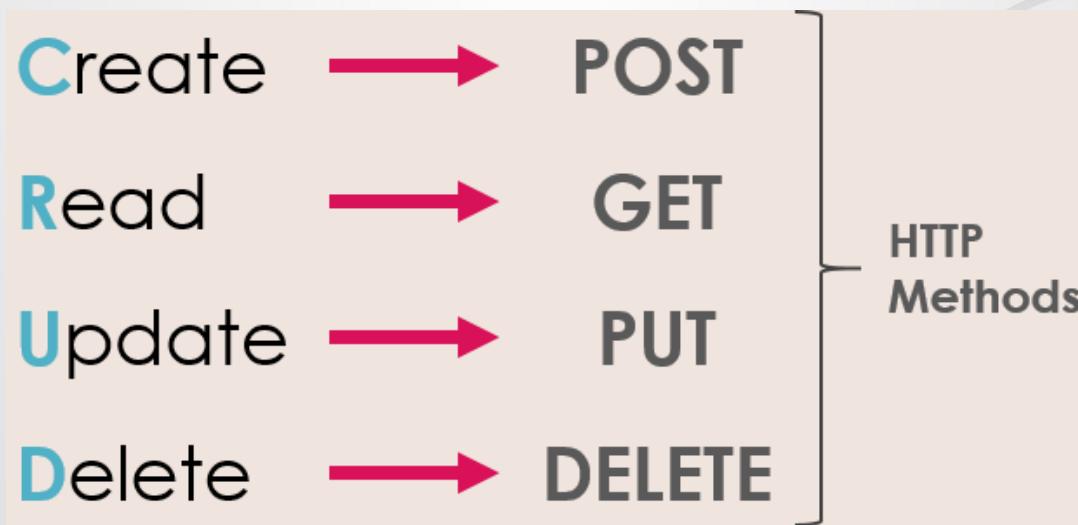
- **Representational State Transfer**

- Es un **estilo arquitectónico** utilizado en el desarrollo de aplicaciones web que se comunican a través de internet.
- Se basa en el protocolo HTTP y utiliza verbos como: **GET, POST, PUT y DELETE**.
- Los recursos se identifican mediante una **URL** y se pueden acceder a través de **HTTP**.
- Las respuestas suelen ser en formato **JSON** o **XML**.
- Promueve la **interoperabilidad** y escalabilidad de las aplicaciones web sin conocer detalles internos de cada uno



REST API

- Se refiere a una API que se adhiere a las restricciones y principios del diseño arquitectónico REST.
- Utiliza los verbos HTTP (GET, POST, PUT, DELETE) para realizar operaciones en los recursos identificados por URLs y devuelve las respuestas en formato JSON o XML.



Desarrollo de ambiente web – Semana 3

HTTP HEADERS



HTTP Headers

- Son campos adicionales que se envían junto con una solicitud o respuesta HTTP.
- Se dividen en dos categorías principales:
 - Request headers
 - Response headers



HTTP Headers

- **User-Agent:** identifica el tipo de agente de usuario (navegador web, robot web, etc.) que está haciendo la solicitud.
- **Accept:** indica los tipos de contenido que el cliente está dispuesto a aceptar. Por ejemplo, "text/html" para contenido HTML o "application/json" para contenido JSON.
- **Authorization:** proporciona credenciales de autenticación para acceder a recursos protegidos en el servidor.
- **Cookie:** incluye una o más cookies almacenadas previamente en el cliente, que se utilizarán para realizar un seguimiento de la sesión.
- **Content-Type:** indica el tipo de contenido de la solicitud. Por ejemplo, "application/json" para datos en formato JSON.

URL: http://intranet/Pages/default.aspx	
Key	Value
Request	GET /Pages/default.aspx HTTP/1.1
Accept	text/html, application/xhtml+xml, */*
Accept-Language	en-ZA
User-Agent	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Accept-Encoding	gzip, deflate
Host	intranet
If-Modified-Since	Fri, 17 Aug 2012 14:52:07 GMT
Connection	Keep-Alive
Cookie	MSOWebPartPage_AnonymousAccessCookie=80; WSS_KeepSessionAuthenticated=



```
1 {  
2   "string": "Hi",  
3   "number": 2.5,  
4   "boolean": true,  
5   "null": null,  
6   "object": { "name": "Kyle", "age": 24 },  
7   "array": ["Hello", 5, false, null, { "key": "value", "number": 6 }],  
8   "arrayOfObjects": [  
9     { "name": "Jerry", "age": 28 },  
10    { "name": "Sally", "age": 26 }  
11  ]  
12 }  
13
```

Desarrollo de ambiente web – Semana 3

HTTP REQUEST METHODS

HTTP Request Methods

Son verbos que se utilizan en una solicitud HTTP para indicar la acción que se desea realizar

- **GET**: solicita una representación del recurso identificado por la URL. El servidor devuelve el contenido del recurso en la respuesta.
- **POST**: envía datos al servidor para crear un nuevo recurso. El contenido de la solicitud se incluye en el cuerpo de la solicitud y el servidor devuelve una respuesta que puede incluir información sobre el recurso recién creado.
- **PUT**: actualiza o reemplaza completamente un recurso identificado por la URL. El contenido de la solicitud se incluye en el cuerpo de la solicitud y el servidor devuelve una respuesta que puede incluir información sobre el recurso actualizado.
- **DELETE**: elimina el recurso identificado por la URL. El servidor devuelve una respuesta que puede incluir información sobre la eliminación del recurso.
- **HEAD**: solicita información sobre el recurso identificado por la URL, pero no solicita su contenido. El servidor devuelve una respuesta que incluye los headers de la respuesta, pero sin el contenido del recurso.
- **OPTIONS**: solicita información sobre los métodos de solicitud HTTP permitidos en la URL. El servidor devuelve una respuesta que incluye los métodos permitidos y otros headers de información.
- **PATCH**: actualiza parcialmente el recurso identificado por la URL. El contenido de la solicitud se incluye en el cuerpo de la solicitud y el servidor devuelve una respuesta que puede incluir información sobre el recurso actualizado.

Desarrollo de ambiente web – Semana 3

HTTP STATUS CODE



HTTP Status Code

Con cada solicitud realizada al servidor, se obtiene códigos de respuesta o estado

- Los códigos de estado se dividen en 5 principales categorías:
 - **1xx** (Respuestas informativas): Indica que la solicitud se ha recibido y se está procesando.
 - **2xx** (Respuestas satisfactorias): Indica que la solicitud se ha procesado correctamente y se ha completado con éxito.
 - **3xx** (Redirecciones): Indica que el cliente debe realizar más acciones para completar la solicitud.
 - **4xx** (Errores del cliente): Indica que la solicitud del cliente no se pudo procesar debido a un error en la solicitud.
 - **5xx** (Errores del servidor): Indica que la solicitud del cliente no se pudo procesar debido a un error en el servidor.

2XX : Success

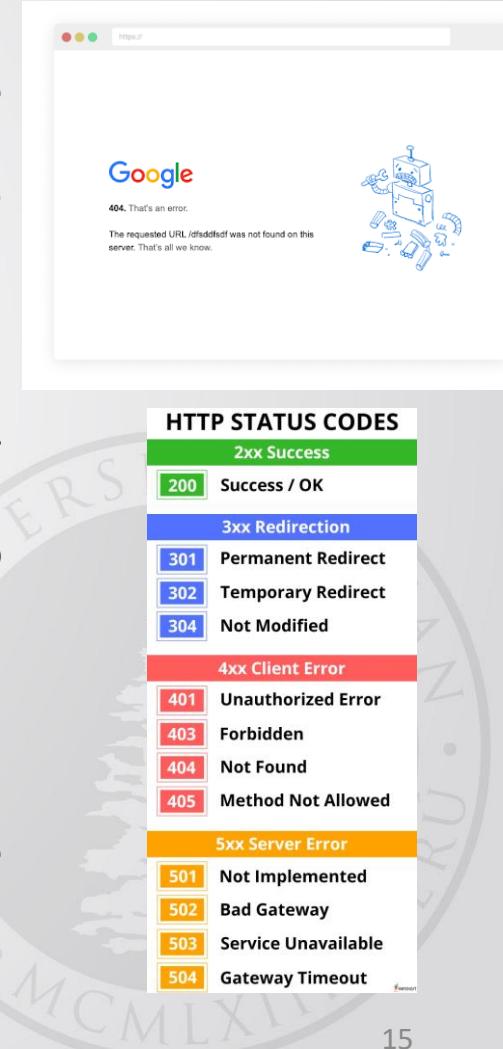
3XX : Redirection

4XX : Problem with the client

5XX : Problem with the server

HTTP Status Code

- Algunos códigos de estado HTTP más comunes:
 - **200 OK**: la solicitud se ha procesado correctamente y se ha completado con éxito.
 - **201 Created**: la solicitud se ha procesado correctamente y se ha creado un nuevo recurso.
 - **400 Bad Request**: la solicitud del cliente no se pudo procesar debido a un error en la solicitud.
 - **401 Unauthorized**: la solicitud del cliente no se pudo procesar porque el cliente no está autorizado a acceder al recurso.
 - **403 Forbidden**: la solicitud del cliente no se pudo procesar porque el servidor se niega a permitir el acceso al recurso.
 - **404 Not Found**: la solicitud del cliente no se pudo procesar porque el servidor no pudo encontrar el recurso solicitado.
 - **500 Internal Server Error**: la solicitud del cliente no se pudo procesar debido a un error en el servidor.



Desarrollo de ambiente web – Semana 3

HTTP ACCESS CONTROL



HTTP Access Control

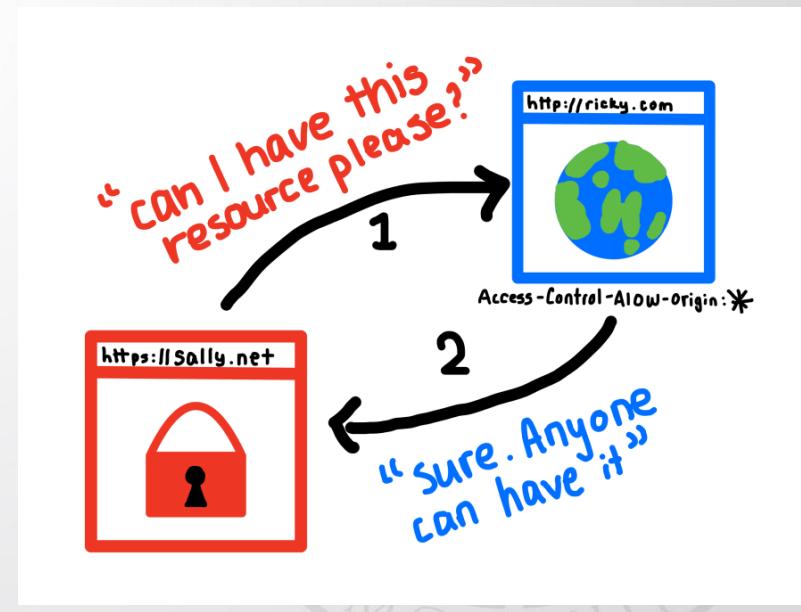
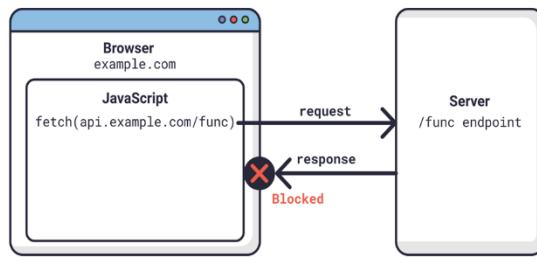
- Es un mecanismo que permite a los servidores web controlar el acceso a sus recursos por parte de los clientes a través de solicitudes HTTP.
- Proporciona seguridad y protección contra el acceso no autorizado a recursos protegidos.



Cross-Origin Resource Sharing (CORS)

- Permite a los servidores web indicar a los navegadores web si se permite que una solicitud de recursos se realice desde un origen diferente al del recurso solicitado.

CORS Explained for Beginners



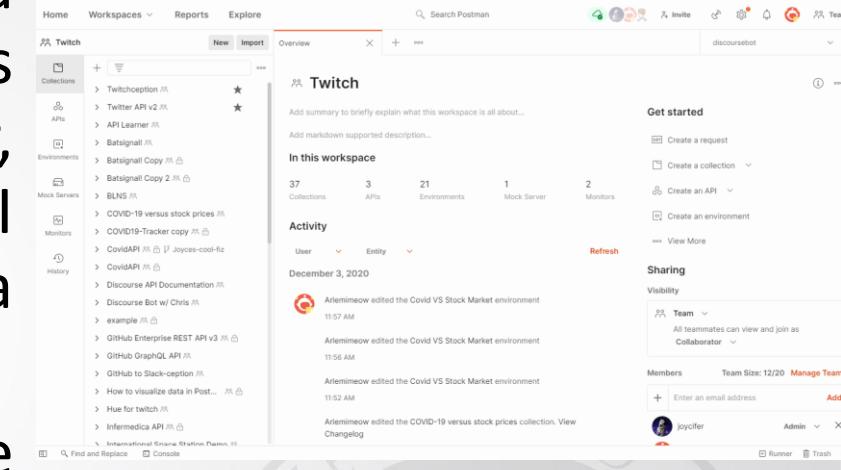
Desarrollo de ambiente web – Semana 3

POSTMAN



POSTMAN

- Es una herramienta de prueba de API que permite a los desarrolladores probar, documentar y compartir las API de sus aplicaciones de una manera fácil y eficiente.
- Cuenta con la capacidad de agregar parámetros, cabeceras y cuerpo a las solicitudes HTTP, la opción de enviar solicitudes a diferentes entornos de API.



Desarrollo de ambiente web – Semana 3

PRUEBAS CON POSTMAN

Pruebas con POSTMAN



POSTMAN
API PLATFORM



Referencias

- Gaitatzis, T. (2019). Learn REST APIs: Your guide to how to find, learn, and connect to the REST APIs that powers the Internet of Things revolution. BackupBrain Press. Chapter 1 Pages 5-9 Chapter 2 Pages 13-21 Chapter 3 29-33 Chapter 4 43-69

Semana 4

Curso:
**Desarrollo de ambiente
web**

Ciclo: 2023-2

**Unidad II: Acceso a datos,
Backend, Web API y Entity
Framework**



UNIDAD DE APRENDIZAJE II:

ACCESO A DATOS, BACKEND, WEB API Y

ENTITY FRAMEWORK

RESULTADOS DE APRENDIZAJE:

- Comprende el ecosistema .NET.
- Entiende la diferencia sobre .NET Framework y .NET Core.
- Comprende sobre la comunicación entre un proyecto web y la base de datos.
- Entiende sobre la importancia de utilizar un ORM.
- Comprende y desarrolla los enfoques de Entity Framework Core.
- Manipula objetos de base de datos mediante Entity Framework Core.
- Entiende la importancia de la programación orientada a objetos y el principio SOLID para la construcción de aplicaciones web empresariales.
- Construye aplicaciones backend con ASP.NET Core Web API.
- Comprende la importancia de los enfoques de Entity Framework Core (Code First y Database First).
- Construye aplicaciones sobre arquitectura limpia (Clean Arquitecture).
- Entiende la importancia de documentar las APIs.
- Entiende la importancia de un middleware en una aplicación web.
- Entiende el uso de procedimientos almacenados en una aplicación web.
- La capacidad de aplicar conocimientos de matemáticas, ciencias e ingeniería en la solución de problemas complejos de ingeniería.
- La capacidad de identificar, formular, buscar información y analizar problemas complejos de ingeniería para llegar a conclusiones fundamentadas usando principios básicos de matemáticas, ciencias naturales y ciencias de la ingeniería.
- La capacidad de comunicarse eficazmente, mediante la comprensión y redacción de informes y documentación de diseño, la realización de exposiciones, y la transmisión y recepción de instrucciones claras.
- La capacidad de comprender y evaluar el impacto de las soluciones a problemas complejos de ingeniería en un contexto global, económico, ambiental y social.
- La capacidad de aplicar el razonamiento informado mediante el conocimiento contextual para evaluar cuestiones sociales, de salud, de seguridad, legales y culturales y las consecuentes responsabilidades relevantes para la práctica profesional de la ingeniería.

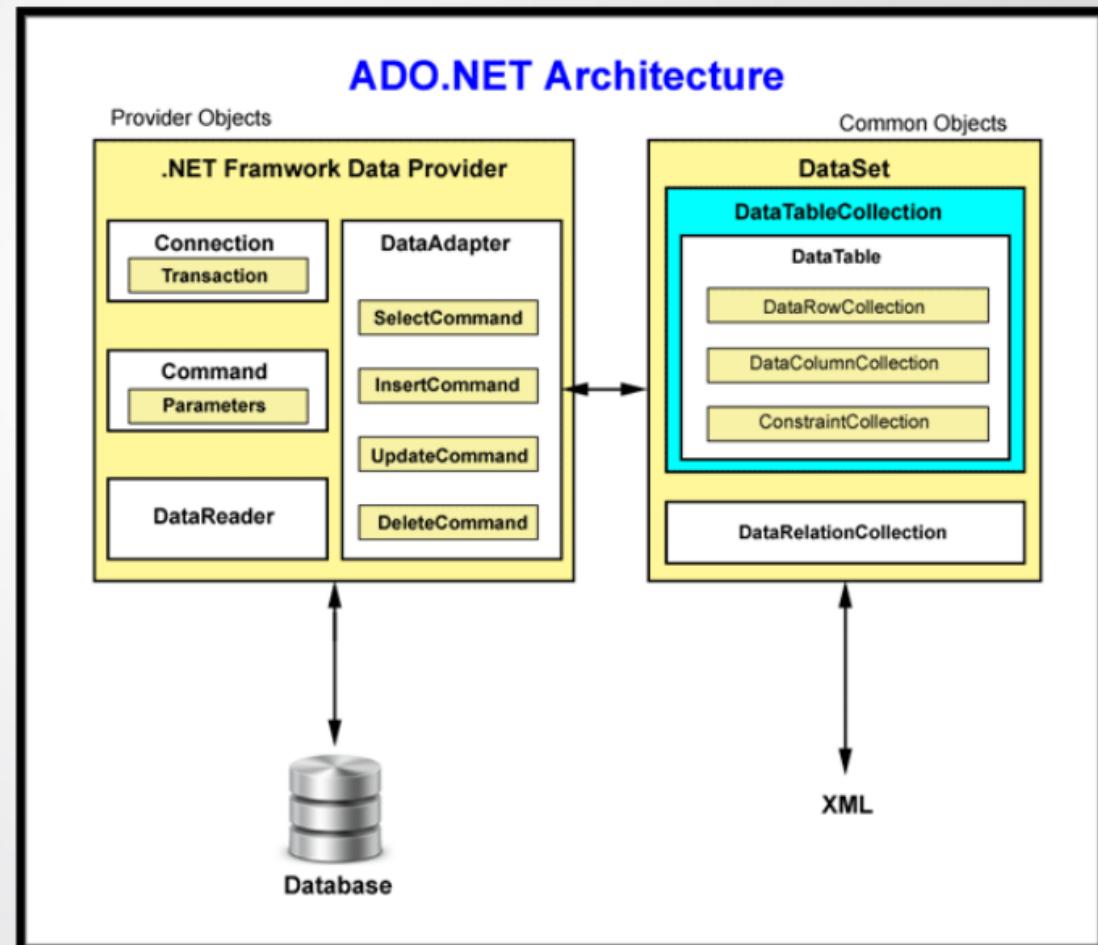
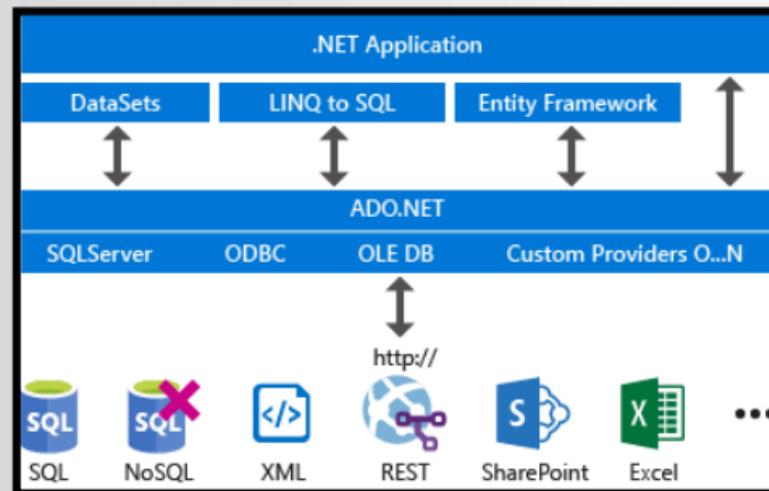
CONTENIDO:

- 2.1. Comprende sobre .NET
- 2.2. ¿Qué es un ORM?
- 2.3. Principios SOLID
- 2.4. Patrones de diseño
- 2.5. Entity Framework Core – Enfoques.
- 2.6. Expresiones lambda y LINQ.

Desarrollo de ambiente web – Semana 4

COMPRENDE SOBRE .NET Y ADO NET

ADO.NET



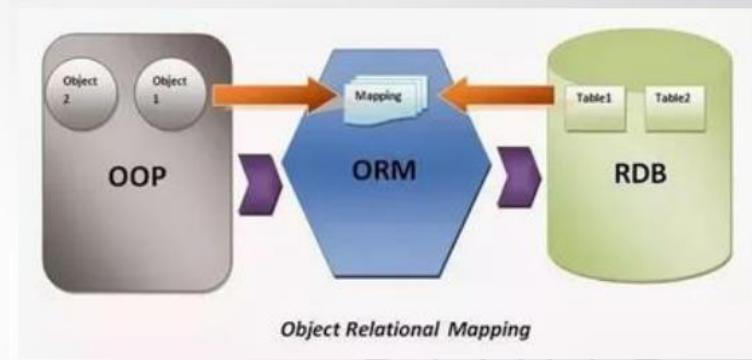
Desarrollo de ambiente web – Semana 4

¿QUÉ ES UN ORM?



Object Relational Mapping

- Es un mecanismo que permite abordar, acceder y manipular objetos (Mapeo) sin tener que considerar cómo los objetos se relacionan con sus fuentes de datos.
- Permite crear una base de datos orientada a objetos “virtual” (Persistencia), sobre una base de datos relacional.
- Se considera una capa intermedia entre la base de datos y los objetos de la base de datos.
- ¿Por qué trabajar con un ORM?: Abstracción de la base de datos, POO, seguridad.



Consultemos a la base de datos..

Consultemos a la base de datos
sin un ORM...

Los primeros 5 registros de la tabla
Clientes



```
SELECT * FROM Clientes WHERE rownum<=5;
```



```
SELECT TOP 5 * FROM Clientes
```



```
SELECT * FROM Clientes LIMIT 5
```



```
SELECT * FROM Clientes fetch first 5 rows only
```

Object Relational Mapping

Consultemos a la base de datos con un ORM...

Los primeros 5 registros de la tabla Clientes.

```
var listado = data.Clientes.Take(5).ToList();
```

Desarrollo de ambiente web – Semana 4

PRINCIPIOS SOLID



Programación Orientada a Objetos: Pilares

```
public class Customer
{
    // Fields, properties, methods and events go here...
}
```

```
Customer object1 = new Customer();
```

```
public class Manager : Employee
{
    // Employee fields, properties, methods and events
    // are inherited
    // New Manager fields, properties, methods and
    // events go here...
}
```

Pilares de la POO

- Abstracción
- Encapsulamiento.
- Herencia.
- Polimorfismo.

Principios SOLID

- **S:** Single responsibility principle o Principio de responsabilidad única. Keyword: “**Decoupled**”
- **O:** Open/closed principle o Principio de abierto/cerrado. Keyword: “**Abstraction**”
- **L:** Liskov substitution principle o Principio de sustitución de Liskov. Keyword: “**Replaceable**”
- **I:** Interface segregation principle o Principio de segregación de la interfaz. Keyword: “**Segregate Interfaces**”.
- **D:** Dependency inversion principle o Principio de inversión de dependencia. Keyword: “**Dependency**”.

Desarrollo de ambiente web – Semana 4

PATRONES DE DISEÑO



Patrones de diseño

- Los patrones de diseño son soluciones habituales a problemas que ocurren con frecuencia en el diseño de software. Son como planos prefabricados que se pueden personalizar para resolver un problema de diseño recurrente en tu código.

¿En qué consiste el patrón?

- El **propósito** del patrón explica brevemente el problema y la solución.
- La **motivación** explica en más detalle el problema y la solución que brinda el patrón.
- La **estructura** de las clases muestra cada una de las partes del patrón y el modo en que se relacionan.
- El **ejemplo de código** en uno de los lenguajes de programación populares facilita la asimilación de la idea que se esconde tras el patrón.

Desarrollo de ambiente web – Semana 4

ENTITY FRAMEWORK CORE - ENFOQUES

Entity Framework Core

- Entity Framework (EF) Core es una versión ligera, extensible, de código abierto y multiplataforma de la popular tecnología de acceso a datos de Entity Framework.
- EF Core puede servir como un mapeador relacional de objetos (ORM), que:
 - Permite a los desarrolladores de .NET trabajar con una base de datos utilizando objetos .NET.
 - Elimina la necesidad de la mayor parte del código de acceso a datos que normalmente es necesario escribir.
- EF Core admite muchos motores de base de datos.
- EF Core Admite
 - Acceder mediante “**Modelos**”.
 - Consultas utilizando **LINQ**.
- **CRUD** a las tablas de base de datos.



Fuente: [Página Web Oficial - Microsoft Entity Framework Core](#)

Desarrollo de ambiente web – Semana 4

EXPRESIONES LAMBDA Y LINQ

Query Expressions & Lambda Expressions

```
// Lista de Estudiantes
IList<Student> studentList = new List<Student>() {
    new Student() { StudentID = 1, StudentName = "Jose", Age = 13} ,
    new Student() { StudentID = 2, StudentName = "Carlos", Age = 21 } ,
    new Student() { StudentID = 3, StudentName = "Bryan", Age = 18 } ,
    new Student() { StudentID = 4, StudentName = "Daniel" , Age = 20} ,
    new Student() { StudentID = 5, StudentName = "Eduardo" , Age = 15 }
};

// Método 1: LINQ Query Expressions para consultar a los estudiantes
// adolescentes
var teenAgerStudent = from s in studentList
where s.Age > 12 && s.Age < 18
select s;

// Método 2: Lambda expression para consultar a los estudiantes
// adolescentes
var teenAgerStudent1 = studentList.Where(x=>x.Age > 12 && x.Age <
18).Select(x=>x).ToList();
```

Referencias

- Price, M. J. (2022). *C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals: Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7, 7th Edition*. Packt Publishing. Chapter 1 Pages 10-24 Chapter 10 425-438 Chapter 10 433-486 Chapter 11 Pages 489-517