# Apunte de Estudio - Curso: Aprende Programación Backend en C# .NET

## Sección 1 - Introducción General

### Objetivo del Curso

Este curso está orientado al **desarrollo de backend** utilizando **C# y .NET**, abordando:

* Conceptos generales de backend aplicables a cualquier lenguaje.
* Fundamentos de .NET y su evolución.
* Desarrollo de APIs REST con ASP.NET.
* Uso del formato JSON para comunicación entre sistemas.

### Perfil del Curso

* Apto tanto para programadores con experiencia como para principiantes.
* Finalidad: comprender y construir **endpoints escalables** que manejen reglas de negocio y recursos.

## .NET: Historia y Estructura

### ¿Qué es un marco de trabajo (framework)?

Es una estructura que define:

* Dónde poner el código.
* Cómo organizarlo.
* Cómo conectarlo entre componentes.

### Evolución de .NET

* **.NET Framework (2002)**: Privado. Competencia directa de Java y PHP.
* **.NET Core (2015)**: Código abierto y multiplataforma. Llega hasta la versión 3.
* **.NET 5+ (2020 en adelante)**: Unificación. Ya no se usa “Core” ni “Framework” en el nombre.
  + Ej.: .NET 5, .NET 6, .NET 7, .NET 8.

### Diferencia entre C# y .NET

* **C#**: Lenguaje de programación.
* **.NET**: Plataforma o framework donde corre C# (entre otros lenguajes).

## ¿Qué es el Backend?

### Definición

* Parte lógica del sistema, no tiene interfaz gráfica.
* Contiene y ejecuta las **reglas de negocio**.
* Atiende solicitudes desde frontend, apps móviles, hardware o videojuegos.

### Accesibilidad

* Está en un **servidor público** accesible mediante URL (IP o dominio).
* Expone **endpoints** accesibles vía **API**.

### Conexiones comunes al backend:

* Frontend (Angular, React, JavaScript, etc.).
* Dispositivos IoT (ej. Arduino, lentes inteligentes).
* Juegos online.

### Reglas de negocio

* Son las condiciones y lógicas específicas de una aplicación.
* Ej.: En Uber, que un auto esté disponible o no, o en MercadoLibre, listar y comprar productos.

## Qué es una API y sus componentes

### API (Application Programming Interface)

Es la interfaz que permite a otras aplicaciones comunicarse con el backend mediante **endpoints**.

### Endpoints

* Son rutas (ej.: /api/login, /api/products).
* Cada uno corresponde a una acción: iniciar sesión, obtener productos, etc.

## Protocolo HTTP

### ¿Qué es HTTP?

* “HyperText Transfer Protocol”: protocolo de transferencia de hipertexto.
* Permite enviar y recibir datos estructurados (como JSON).

### Verbos HTTP

* **GET**: Obtener datos.
* **POST**: Enviar nuevos datos.
* **PUT**: Actualizar datos.
* **DELETE**: Eliminar datos.
* **PATCH**: Actualización parcial (no siempre utilizado).

### Códigos de respuesta HTTP

* **1xx**: Informativos.
* **2xx**: Éxito (ej.: 200 OK).
* **3xx**: Redirecciones.
* **4xx**: Errores del cliente (ej.: 400 Bad Request).
* **5xx**: Errores del servidor (ej.: 500 Internal Server Error).

Los más comunes en el curso:

* **200**: Todo correcto.
* **400**: Datos enviados incorrectos.
* **500**: Error interno del servidor.

## Estructura de una Solicitud y una Respuesta HTTP

### Solicitud HTTP (Request)

1. **URL**: La ruta del recurso solicitado.
2. **Verbo (método)**: GET, POST, etc.
3. **Headers**: Información adicional (tipo de contenido, idioma, token de autorización).
4. **Body**: Solo en POST/PUT. Incluye datos en formato JSON, XML u otros.

Ejemplo de body JSON:

{  
 "usuario": "juan",  
 "password": "12345"  
}

### Respuesta HTTP (Response)

1. **Código de estado**: Indica el resultado (200, 400, 500, etc.).
2. **Headers**: Tipo de contenido (HTML, JSON, etc.), idioma, cookies, etc.
3. **Body**: Contenido devuelto, usualmente en formato JSON.

## Introducción al Formato JSON

### ¿Qué es JSON?

* “JavaScript Object Notation”.
* Formato de texto legible para intercambiar datos entre sistemas.
* No depende del lenguaje de programación.

### Ejemplo de Objeto JSON

{  
 "total": 100.50,  
 "customer": "Pedro",  
 "paid": true  
}

### Objetos Anidados y Colecciones

* Se pueden tener objetos dentro de otros objetos:

"address": {  
 "street": "Calle Juárez",  
 "number": 123,  
 "cp": 55235  
}

* Se pueden tener listas (arrays) de objetos:

"items": [  
 {  
 "description": "papas",  
 "amount": 1.0,  
 "quantity": 5  
 },  
 {  
 "description": "gaseosa",  
 "amount": 2.5,  
 "quantity": 2  
 }  
]

## Conclusión de la Sección

Con esta introducción, se sientan las bases para el desarrollo backend:

* Comprensión de .NET y C#.
* Uso de APIs y protocolos HTTP.
* Dominio de JSON como formato de intercambio de datos.

El conocimiento de estas bases es esencial para construir servicios backend robustos, reutilizables y escalables.