Perfecto ✅  
Con todo el material de la sección que me diste (tanto las transcripciones que ya me pasaste en el chat como el archivo que contiene las últimas clases), te armo un **resumen unificado** para que puedas repasar y volver a entender todos los temas de esta parte del curso.

**Resumen Sección – ASP.NET Core Web API (Controladores, Rutas, Métodos HTTP y Tipos de Respuesta)**

**1. Creación del Proyecto y Controlador Base**

* Se crea el proyecto en **Visual Studio 2022** seleccionando **ASP.NET Core Web App (C#)**.
* Evitar carpetas sincronizadas con nubes (OneDrive, Google Drive) para evitar lentitud.
* Usar .NET 8 (funciona en .NET 5+ en la mayoría de casos).
* Activar **Usar controladores**.
* Al ejecutar, Swagger documenta automáticamente la API.
* Ejemplo inicial WeatherForecast para entender la estructura.

**2. Concepto de Controlador**

* Un **controlador** es una clase que **hereda de ControllerBase** y actúa como punto de entrada (endpoint) para recibir solicitudes HTTP y devolver respuestas (números, cadenas, HTML, JSON, XML, etc.).
* Convención de rutas:
* https://localhost:puerto/api/[NombreControladorSinController]
* Métodos dentro del controlador son mapeados a endpoints mediante atributos como [HttpGet].

**3. Métodos HTTP (Verbos)**

* **Principales:** GET, POST, PUT, DELETE.
* [HttpGet] → Obtener datos.
* [HttpPost] → Insertar/crear datos.
* [HttpPut] → Actualizar datos.
* [HttpDelete] → Eliminar datos.
* Ejemplo: un mismo controlador puede tener métodos distintos con diferentes verbos para operaciones aritméticas (suma, resta, multiplicación, división).
* **Swagger** permite probar rápidamente, **Postman** es más completo y realista para entornos de desarrollo.

**4. Parámetros en URL vs Body**

* Parámetros en URL (?a=10&b=22) → comunes en **GET**.
* Parámetros en **Body** (JSON) → comunes en **POST** y **PUT**.
* ASP.NET Core **deserializa automáticamente** el JSON recibido en el body a un objeto de C# si el método recibe un parámetro de clase.
* Ejemplo:
* public IActionResult Add([FromBody] Numbers n) { ... }

**5. Headers (Encabezados)**

* Contienen información adicional: origen, cliente, autenticación, tipo de contenido, etc.
* Se accede con [FromHeader]:
  + Nombre exacto → mapeo automático.
  + Con guiones → [FromHeader(Name = "Content-Length")].
* **Headers personalizados** → convención X-... (ej. X-Sum).
* Útiles para datos adicionales fuera del body y la URL.

**6. Tipos de Respuesta con Objetos**

* Un endpoint puede devolver no solo tipos simples (string, int) sino objetos o listas.
* Ejemplo: devolver lista de People desde un repositorio en memoria (static List<People>).
* Filtrado de datos:
  + Por ID usando First(...).
  + Por búsqueda parcial (Contains) con Where(...) y manejo de mayúsculas/minúsculas (ToUpper).
* Mala práctica: devolver todos los registros sin paginación ni filtro en datos grandes.

**7. Tipos de Respuesta Enriquecidos**

* **ActionResult<T>** → permite devolver datos junto con códigos HTTP adecuados.
  + Ejemplo: NotFound() (404), Ok(obj) (200).
  + Útil para manejar casos donde el recurso no existe sin que el servicio “truene”.
* **IActionResult** → para casos donde no se devuelve contenido, solo el código.
  + Ejemplo: NoContent() (204) para operaciones exitosas sin body.
  + Validaciones → BadRequest() (400) si los datos son inválidos.
* Ventaja: mayor control sobre la respuesta y claridad semántica (los métodos como Ok, NotFound, BadRequest, etc. son autoexplicativos).

**Mapa mental simplificado**

Proyecto ASP.NET Core

├── Controladores (heredan ControllerBase)

│ ├── Métodos HTTP → GET, POST, PUT, DELETE

│ ├── Parámetros en URL

│ ├── Parámetros en Body (JSON → clase C#)

│ └── Headers (estándar y personalizados)

│

├── Tipos de Respuesta

│ ├── Datos simples → string, int, decimal

│ ├── Objetos/Listas → List<People>

│ ├── Filtrado → First, Where, Contains

│ ├── ActionResult<T> → Datos + Código HTTP

│ └── IActionResult → Código HTTP sin contenido

│

└── Herramientas

├── Swagger → documentación y pruebas rápidas

└── Postman → pruebas avanzadas, colecciones