ATRIBUTOS EF C#  
Debido a que la cantidad de atributos es muy extensa y la creación de un "canvas" como tal no es posible, te proporciono una lista exhaustiva y organizada de los atributos más relevantes de .NET para mapeo y validación de datos.

He dividido la información en secciones claras para que puedas imprimirla y usarla como referencia.

# Atributos de Mapeo de Datos (Entity Framework Core)

Estos atributos (System.ComponentModel.DataAnnotations.Schema y Microsoft.EntityFrameworkCore) son los que Entity Framework (EF) utiliza para definir cómo tus clases se relacionan con el esquema de la base de datos.

* [Table("NombreDeTabla")]: Especifica el nombre de la tabla en la base de datos si es diferente al de la clase.
  + Ejemplo: [Table("Productos")]
* [Column("NombreColumna", TypeName = "sql\_type")]: Especifica el nombre y tipo de la columna en la base de datos.
  + Ejemplo: [Column("Precio", TypeName = "decimal(18, 2)")]
* [Key]: Define la clave primaria de la entidad. No es necesario si la propiedad se llama **Id** o <NombreDeClase>Id.
  + Ejemplo: [Key] public int ProductoId { get; set; }
* [ForeignKey("NombrePropiedad")]: Define la clave foránea para la relación. Se usa en la propiedad de navegación o en la propiedad de la clave foránea misma.
  + Ejemplo: [ForeignKey("CategoriaId")] public Categoria Categoria { get; set; }
* [InverseProperty("NombrePropiedadDeNavegacion")]: Se usa para especificar la propiedad de navegación opuesta en una relación uno a uno o uno a muchos, ayudando a EF a comprender la relación.
  + Ejemplo: [InverseProperty("Producto")] public ICollection<DetalleOrden> Detalles { get; set; }
* [NotMapped]: Excluye una propiedad de ser mapeada a una columna en la base de datos, lo que es útil para propiedades de solo lógica.
  + Ejemplo: [NotMapped] public string NombreCompleto { get; set; }
* [DatabaseGenerated(DatabaseGeneratedOption)]: Especifica cómo la base de datos genera el valor de la columna (ej. Identity para auto-incremento, Computed para un valor calculado).
  + Ejemplo: [DatabaseGenerated(DatabaseGeneratedOption.Identity)] public int Id { get; set; }
* [StringLength(maxLength)]: Limita la longitud máxima de una propiedad de tipo string, lo que se traduce en un VARCHAR(maxLength) en SQL.
  + Ejemplo: [StringLength(255)] public string Descripcion { get; set; }
* [Precision(precision, scale)]: Define la precisión y escala para un tipo numérico o decimal, como DECIMAL(8, 2) para un número con 8 dígitos en total y 2 decimales.
  + Ejemplo: [Precision(18, 2)] public decimal Precio { get; set; }
* [Required]: Indica que una columna no puede ser nula. EF la mapea a un campo NOT NULL.
  + Ejemplo: [Required] public string Nombre { get; set; }
* [Index(unique: true)]: Crea un índice sobre la columna o columnas especificadas para mejorar el rendimiento de las consultas. El parámetro unique: true asegura que los valores en esa columna sean únicos.
  + Ejemplo: [Index(unique: true)] public string CodigoSKU { get; set; }
* [DefaultValueSql("GETDATE()")]: Permite establecer un valor predeterminado para una columna usando una función SQL.
  + Ejemplo: [DefaultValueSql("GETDATE()")] public DateTime FechaCreacion { get; set; }

# Atributos de Validación (Data Annotations)

Estos atributos (System.ComponentModel.DataAnnotations) se usan para validar los datos en el modelo, tanto en el frontend como en el backend. Son una forma declarativa de asegurar que los datos cumplen con ciertas reglas.

* [Required]: Indica que un campo es obligatorio.
  + Ejemplo: [Required] public string NombreUsuario { get; set; }
* [StringLength(maxLength, MinimumLength = minLength)]: Define un rango de longitud para una cadena.
  + Ejemplo: [StringLength(50, MinimumLength = 5)] public string Titulo { get; set; }
* [MinLength(minLength)]: Define la longitud mínima de una cadena.
  + Ejemplo: [MinLength(5)] public string Comentario { get; set; }
* [MaxLength(maxLength)]: Define la longitud máxima de una cadena.
  + Ejemplo: [MaxLength(50)] public string Etiqueta { get; set; }
* [Range(min, max)]: Valida que el valor numérico esté dentro de un rango.
  + Ejemplo: [Range(18, 99)] public int Edad { get; set; }
* [RegularExpression("regex")]: Valida la propiedad contra una expresión regular.
  + Ejemplo: [RegularExpression(@"^[0-9]{5}$")] public string CodigoPostal { get; set; }
* [EmailAddress]: Valida que el formato del string sea el de un correo electrónico.
  + Ejemplo: [EmailAddress] public string CorreoElectronico { get; set; }
* [Phone]: Valida que el formato del string sea el de un número de teléfono.
  + Ejemplo: [Phone] public string Telefono { get; set; }
* [Url]: Valida que el formato del string sea el de una URL.
  + Ejemplo: [Url] public string SitioWeb { get; set; }
* [Compare("OtraPropiedad")]: Se usa para comparar dos propiedades, como al confirmar una contraseña.
  + Ejemplo: [Compare("Password")] public string ConfirmarPassword { get; set; }
* [CreditCard]: Valida el formato de un número de tarjeta de crédito.
  + Ejemplo: [CreditCard] public string NumeroTarjeta { get; set; }
* [FileExtensions(Extensions = "ext1,ext2")]: Valida que la extensión de un archivo sea una de las permitidas.
  + Ejemplo: [FileExtensions(Extensions = "jpg,png,gif")] public string FotoPerfil { get; set; }
* [DataType(DataType.Password)]: Ayuda a los frameworks (como ASP.NET Core) a renderizar la UI adecuada, por ejemplo, mostrando un campo de contraseña.
  + Ejemplo: [DataType(DataType.Password)] public string Password { get; set; }

# Atributos para la Serialización de JSON

Estos atributos (System.Text.Json.Serialization) controlan cómo las propiedades de tus objetos se serializan y deserializan a JSON. Son muy importantes para las APIs.

* [JsonPropertyName("nombrePropiedadJson")]: Especifica el nombre de la propiedad en el JSON si es diferente al de la clase de C#.
  + Ejemplo: [JsonPropertyName("producto\_id")] public int Id { get; set; }
* [JsonIgnore]: Ignora una propiedad durante la serialización, útil para no exponer datos sensibles.
  + Ejemplo: [JsonIgnore] public string ClaveApi { get; set; }
* [JsonInclude]: Incluye una propiedad durante la serialización que de otro modo sería ignorada (por ejemplo, si es un campo privado o una propiedad de solo lectura).
  + Ejemplo: [JsonInclude] private readonly int \_nivelSeguridad;
* [JsonExtensionData]: Permite manejar propiedades JSON adicionales que no están definidas en la clase.
  + Ejemplo: [JsonExtensionData] public Dictionary<string, object> PropiedadesAdicionales { get; set; }
* [JsonNumberHandling(options)]: Controla cómo se manejan los números durante la serialización, por ejemplo, si los números en formato string deben ser aceptados.
  + Ejemplo: [JsonNumberHandling(JsonNumberHandling.AllowReadingFromString)] public int Valor { get; set; }
* [JsonConverter(typeof(MiClaseConverter))]: Permite especificar un convertidor personalizado para una propiedad, lo que te da un control total sobre cómo se serializa y deserializa.
  + Ejemplo: [JsonConverter(typeof(DateTimeOffsetConverter))] public DateTimeOffset UltimaActualizacion { get; set; }
* [JsonIgnoreCondition(JsonIgnoreCondition.WhenWritingDefault)]: Ignora la propiedad si su valor es el valor predeterminado del tipo al escribir el JSON (ej. 0 para enteros o null para objetos).
  + Ejemplo: [JsonIgnoreCondition(JsonIgnoreCondition.WhenWritingDefault)] public int Cantidad { get; set; }