

Podrá descargar algunos elementos de este libro en la página web de Ediciones ENI: <http://www.ediciones-eni.com>.  
Escriba la referencia ENI del libro **RIT10CSHAVSC** en la zona de búsqueda y valide. Haga clic en el título y después en el botón de descarga.

## Prólogo

### Capítulo 1

#### Introducción

1. ¿Qué es C#? .....	9
1.1 ¿Qué se puede hacer con C#? .....	10
1.2 ¿El lenguaje es estable y permanente? .....	12
2. Preparar el entorno .....	13
2.1 Instalación y configuración de Visual Studio Code .....	14
2.2 Instalar las herramientas de compilación .....	16
3. ¿Cómo funciona C#? .....	19

### Capítulo 2

#### Primer programa

1. Crear la primera aplicación C# .....	23
2. Comprender y escribir código C# .....	27
2.1 Conceptos de variable y constante .....	29
2.1.1 Tipos numéricos .....	31
2.1.2 Tipos textuales .....	33
2.1.3 Valor booleano .....	35
2.1.4 Operadores .....	35
2.2 Otros tipos .....	38
2.2.1 Almacenamiento de las fechas .....	38
2.2.2 Intervalos de tiempo .....	40

# 2 \_\_\_\_\_ C# 10 y Visual Studio Code

Fundamentos del lenguaje

3.	Analizar la estructura de un proyecto C#	41
3.1	El concepto de bloques	42
3.2	Significado de los bloques de código	44
3.2.1	El bloque de espacio de nombres	45
3.2.2	Definición de una clase	47
3.2.3	Definición de un método	48
3.3	Declaración «top-level»	48
4.	Ejecutar un programa C#	49
4.1	Lanzar el programa con Visual Studio Code	49
4.2	Lanzar desde la línea de comando	51
5.	Ejercicio	53
5.1	Enunciado	53
5.2	Solución	54

## Capítulo 3

### Programación orientada a objetos

1.	Principios de la programación orientada a objetos	55
1.1	¿Qué es una clase?	55
1.1.1	Las clases en Visual Studio Code	57
1.1.2	Herencia	58
1.1.3	Encapsulación	59
1.2	¿Qué se puede declarar dentro de una clase?	60
1.2.1	Métodos	60
1.2.2	Declarar un dato	63
1.3	Instanciar una clase	67
1.3.1	El constructor	67
1.3.2	Instanciación con la palabra clave new	69
1.4	Polimorfismo	71

2.	Conceptos avanzados . . . . .	73
2.1	Herencia avanzada . . . . .	73
2.1.1	Métodos virtuales . . . . .	73
2.1.2	Clase abstracta . . . . .	74
2.1.3	Interfaz . . . . .	76
2.1.4	Implementación predeterminada en una interfaz . . . . .	78
2.1.5	Enmascaramiento . . . . .	79
2.1.6	Prohibir la herencia . . . . .	80
2.2	Los diferentes tipos de objetos . . . . .	81
2.2.1	Tipos de referencia . . . . .	81
2.2.2	Tipos de valor . . . . .	82
2.2.3	Tipos que aceptan valores null . . . . .	85
2.2.4	Tipos de referencia que aceptan valores NULL . . . . .	86
2.2.5	Las enumeraciones . . . . .	88
2.2.6	Registros . . . . .	89
2.3	Modificadores de clase . . . . .	91
2.3.1	El concepto de static . . . . .	92
2.3.2	El concepto de clase parcial . . . . .	93
3.	Ejercicio . . . . .	94
3.1	Enunciado . . . . .	94
3.2	Solución . . . . .	95

**Capítulo 4**  
**Algoritmia**

1.	Bases de algoritmia . . . . .	99
1.1	Lógica condicional . . . . .	99
1.1.1	Prueba simple: el if/else . . . . .	100
1.1.2	Pruebas múltiples con la instrucción switch . . . . .	106
1.1.3	Coincidencia de patrones . . . . .	108
1.1.4	Ejercicio - enunciado . . . . .	113
1.1.5	Ejercicio - solución . . . . .	113

# 4 \_\_\_\_\_ C# 10 y Visual Studio Code

Fundamentos del lenguaje

1.2	Las colecciones	114
1.2.1	La interfaz IEnumerable	114
1.2.2	Las tablas	115
1.2.3	La lista	117
1.2.4	Los diccionarios	120
1.2.5	Las colecciones algorítmicas	122
1.3	Los bucles	124
1.3.1	Información general sobre los bucles	124
1.3.2	El bucle for	125
1.3.3	El bucle while	126
1.3.4	El bucle do while	127
1.3.5	El bucle foreach	127
1.3.6	La palabra clave yield	128
1.3.7	Ejercicio - enunciado	129
1.3.8	Ejercicio - solución	130
2.	Gestión de los errores	132
2.1	Concepto de una excepción	133
2.2	Devolver una excepción	134
2.3	Gestionar una excepción	136
2.3.1	Bloques try, catch y finally	136
2.3.2	Filtro en bloque catch	138
2.4	Excepciones y rendimientos	141

## Capítulo 5

### LINQ

1.	Funcionamiento básico	143
2.	Variables anónimas	146
3.	Principios de los operadores LINQ	146
3.1	Operadores de producción	150
3.2	Operadores de selección	161
3.3	Operadores de generación	166

- 4. Expresión de consulta LINQ ..... 167
  - 4.1 La palabra clave into ..... 168
  - 4.2 La palabra clave let ..... 170
- 5. Ejercicio ..... 170
  - 5.1 Enunciado ..... 170
  - 5.2 Solución ..... 171

**Capítulo 6**  
**Serialización**

- 1. Serialización en C# ..... 173
- 2. Serialización binaria ..... 174
  - 2.1 Uso de los atributos ..... 175
  - 2.2 Uso de la interfaz ISerializable ..... 179
- 3. Serialización XML ..... 181
  - 3.1 XmlSerializer ..... 181
  - 3.2 XmlDocument, XElement y XAttribute ..... 184
- 4. Serialización JSON ..... 188
  - 4.1 Utf8JsonReader y Utf8JsonWriter ..... 189
  - 4.2 JsonDocument ..... 192
  - 4.3 JsonSerializer ..... 193
- 5. Ejercicio ..... 197
  - 5.1 Enunciado ..... 197
  - 5.2 Solución ..... 199

# 6 \_\_\_\_\_ C# 10 y Visual Studio Code

Fundamentos del lenguaje

## Capítulo 7 Conceptos avanzados

1. Asincronismo .....	201
1.1 Funcionamiento básico .....	201
1.2 Thread y asincronismo.....	203
1.3 Asincronismo en C# .....	204
1.4 Las palabras clave async y await .....	205
1.5 Flujos asíncronos .....	208
2. Algoritmia avanzada .....	210
2.1 Programación dirigida por eventos .....	210
2.1.1 Los delegate .....	210
2.1.2 Los eventos .....	212
2.2 Tipos genéricos .....	215
2.2.1 Uso estándar .....	215
2.2.2 Limitaciones en el tipo genérico .....	217
2.3 Gestión de la memoria .....	218
2.3.1 El destructor .....	219
2.3.2 IDisposable e IAsyncDisposable .....	220
2.4 Parámetros de métodos avanzados .....	221
2.4.1 Parámetro opcional.....	221
2.4.2 Palabras clave de parámetros .....	222
2.4.3 Denominación de parámetros .....	225
2.4.4 Parámetros variables.....	226
2.5 Extensión del funcionamiento de un tipo.....	226
2.5.1 Métodos de extensión .....	227
2.5.2 Definición de los operadores .....	228
2.6 Tuplas y deconstrucción .....	231
2.6.1 Las tuplas en C# 7 .....	231
2.6.2 Deconstrucción de tipo .....	233
2.7 Función local.....	236

**Capítulo 8**  
**Crear aplicaciones**

- 1. Aplicación web ..... 239
  - 1.1 Aplicaciones web gráficas. .... 239
    - 1.1.1 ASP.NET MVC. .... 240
    - 1.1.2 ASP.NET Razor Pages ..... 245
    - 1.1.3 Blazor. .... 250
  - 1.2 API. .... 254
- 2. Aplicación de escritorio ..... 260
  - 2.1 WinForms ..... 261
  - 2.2 Windows Presentation Foundation (WPF) ..... 267
  - 2.3 Universal Windows Platform (UWP) ..... 271
- 3. Aplicación móvil. .... 276
  - 3.1 Instalación. .... 277
    - 3.1.1 Instalación desde la línea de comando ..... 277
    - 3.1.2 Instalación con Visual Studio 2022 ..... 281
  - 3.2 Código ..... 286
- 4. Conclusión ..... 288

**Capítulo 9**  
**Referencia**

- 1. Introducción ..... 289
- 2. Palabras clave de tipo ..... 289
- 3. Palabras clave de programación orientada a objetos ..... 291
- 4. Palabras clave algorítmicas. .... 295

Índice ..... 301

Podrá descargar algunos elementos de este libro en la página web de Ediciones ENI: <http://www.ediciones-eni.com>.  
Escriba la referencia ENI del libro **EIT2ECR** en la zona de búsqueda y valide. Haga clic en el título y después en el botón de descarga.

## Capítulo 1 Introducción

1. Prólogo . . . . .	9
2. Los riesgos de la sobreoptimización . . . . .	10
3. El principio del perfilado y el benchmarking . . . . .	12
4. El reto financiero y ecológico . . . . .	15
5. Metodología . . . . .	17
6. Entorno necesario . . . . .	18

## Capítulo 2 Principios de perfilado

1. Una actividad estrictamente regulada . . . . .	19
2. Estabilidad de la plataforma . . . . .	25
2.1 ¿Por qué esta regla? . . . . .	25
2.2 ¿Cómo aplicar esta regla? . . . . .	26
2.3 Extraer los casos particulares . . . . .	27
2.4 Relación con la mantenibilidad . . . . .	28
3. Neutralidad del entorno . . . . .	30
3.1 El principio . . . . .	30
3.2 Virtualización . . . . .	31
3.2.1 Virtualización pesada . . . . .	31
3.2.2 Virtualización ligera . . . . .	32
3.3 Transferencia de la visualización . . . . .	33
3.4 Efectos de cachés externas . . . . .	34
3.5 Procesos externos . . . . .	36



# 2 \_\_\_\_\_ Escribir código .NET eficaz

Perfilado, benchmarking y buenas prácticas

3.6	Servicios diversos	37
3.7	Comparación con la nanotecnología	37
4.	Establecer objetivos antes del análisis	38
5.	Mejoras cuantificables	42
5.1	¿Por qué esta regla?	42
5.2	¿Cómo aplicar esta regla?	42
6.	Granularidad descendente	43
6.1	Empecemos con una parábola	43
6.2	Un ejemplo	44
6.3	Advertencia	45
6.4	¿Quién es responsable?	47

## Capítulo 3

### Perfilar una aplicación .NET

1.	Gestión de la memoria de la parte de .NET	49
1.1	Principios de base	49
1.2	Gestión automatizada de memoria y rendimiento	50
1.3	El caso particular del tiempo real	50
1.3.1	Aclarar un malentendido	50
1.3.2	El no determinismo de los recolectores de basura	52
1.4	Asignar memoria	55
1.5	Cómo nos ayuda .NET	61
1.6	Tipos por valores y referencias	62
1.6.1	Funcionamiento de una pila	63
1.6.2	Regresar al código IL	69
1.6.3	Diferencia entre valor y referencia	71
1.6.4	El caso particular de las estructuras	76
1.6.5	Vincular al rendimiento	78
1.7	Calcular el tamaño de la memoria a asignar	79
1.7.1	El caso del código gestionado al 100 %	79
1.7.2	El caso de objetos sujetos a interoperabilidad	80

1.8	Recolección de la memoria . . . . .	82
1.8.1	Criterios para la recolección de la memoria . . . . .	82
1.8.2	Mecanismo de compactación . . . . .	84
1.8.3	Memoria fija y fragmentación . . . . .	85
1.8.4	Activar y ejecutar el recolector de basura . . . . .	87
1.8.5	Noción de generación . . . . .	88
1.8.6	Impacto de la codificación en el rendimiento . . . . .	89
1.8.7	Elegir el recolector de basura . . . . .	93
1.9	Boxing, unboxing y el rendimiento asociado . . . . .	94
1.9.1	¿Cuál es el problema? . . . . .	94
1.9.2	El boxing/unboxing y el rendimiento . . . . .	99
1.9.3	El remedio . . . . .	101
1.10	Gestión de la memoria y finalización . . . . .	105
1.10.1	¡Dejemos que .NET lo haga! . . . . .	105
1.10.2	Liberar los recursos externos durante la ejecución del GC . . . . .	106
1.10.3	Funcionamiento del finalizador . . . . .	110
1.10.4	Liberar los recursos lo antes posible . . . . .	111
1.10.5	Combinar las dos operaciones . . . . .	114
1.11	Una observación final . . . . .	118
2.	Particularidades de las funciones en línea . . . . .	119
2.1	Mecanismo de las funciones en línea . . . . .	119
2.2	Problemática de rendimiento y funciones en línea (online) . .	123
2.3	Impacto en los perfiladores . . . . .	127
3.	Impacto de la gestión de la memoria en el rendimiento . . . . .	130
3.1	Una gran diversidad en los impactos . . . . .	130
3.2	Uso de la memoria virtual . . . . .	130
3.3	Fugas de memoria . . . . .	134
3.4	Montón especial para los objetos grandes y la fragmentación de memoria . . . . .	141
4.	Otros recursos que hay que vigilar . . . . .	144
4.1	La memoria no lo es todo . . . . .	144
4.2	La CPU . . . . .	145

# 4 \_\_\_\_\_ Escribir código .NET eficaz

Perfilado, benchmarking y buenas prácticas

4.3	Las entradas/salidas . . . . .	148
4.4	Espacio de disco disponible . . . . .	150
4.5	Ancho de banda . . . . .	151

## Capítulo 4

### Aplicación de prueba

1.	Preámbulo . . . . .	153
1.1	Una migración histórica . . . . .	153
2.	Criterios de elección . . . . .	154
2.1	¿Por qué esta aplicación? . . . . .	154
2.2	Utilizar la retroalimentación . . . . .	155
2.3	La elección de la transparencia . . . . .	156
2.4	Los límites de la transparencia . . . . .	157
3.	Aplicación seleccionada . . . . .	158
3.1	Campo de uso . . . . .	158
3.2	Arquitectura . . . . .	158
3.3	Interfaz . . . . .	158
3.4	Descripción de la actividad . . . . .	159
4.	Detalles de la aplicación . . . . .	159
4.1	Encontrar la aplicación . . . . .	159
4.2	Instalar la base de datos . . . . .	160
4.2.1	Crear manualmente . . . . .	161
4.2.2	Usar los scripts de generación . . . . .	162
4.3	Instalar la aplicación . . . . .	162
4.3.1	Abrir la solución . . . . .	162
4.3.2	Configurar el proyecto . . . . .	163
4.3.3	Ejecutar la aplicación . . . . .	164
4.4	Detalles de los ensamblés . . . . .	164
4.5	Arquitectura del cliente . . . . .	165
4.6	Estructura de los servicios web . . . . .	165
4.7	Estructura de la base de datos . . . . .	166

- 5. Explicación sobre la complejidad de la aplicación . . . . . 166
- 6. Método recomendado. . . . . 167

**Capítulo 5**  
**Presentación de las herramientas**

- 1. Elegir las herramientas . . . . . 171
- 2. Visual Studio 2022 . . . . . 172
  - 2.1 Ventana de diagnóstico . . . . . 173
  - 2.2 Sesión de perfilado . . . . . 175
- 3. Contadores de rendimiento . . . . . 179
  - 3.1 Terminología. . . . . 179
  - 3.2 Windows . . . . . 180
  - 3.3 Linux y macOS . . . . . 184
- 4. BenchmarkdotNet . . . . . 186
  - 4.1 Crear un proyecto de benchmark . . . . . 187
  - 4.2 Crear un benchmark. . . . . 187
  - 4.3 Ejecutar el benchmark . . . . . 189
- 5. Herramientas alternativas . . . . . 190

**Capítulo 6**  
**Perfilado**

- 1. ¿Por dónde empezar? . . . . . 191
- 2. Escenarios de perfilado . . . . . 192
  - 2.1 Primer escenario: mostrar los datos de una persona. . . . . 192
  - 2.2 Segundo escenario: mostrar y editar un contrato. . . . . 195
- 3. Ejecutar un perfilado . . . . . 196
  - 3.1 Perfilar la API: primer escenario. . . . . 197
    - 3.1.1 Ejecutar con el perfilador . . . . . 199
    - 3.1.2 Primera optimización . . . . . 204
    - 3.1.3 Prueba de carga . . . . . 207

# 6 **Escribir código .NET eficaz**

Perfilado, benchmarking y buenas prácticas

3.1.4	Segunda optimización	213
3.2	Refactorizar tras las optimizaciones	216
3.2.1	Mezcla de responsabilidades	217
3.2.2	Suprimir static	218
3.2.3	Usar la inyección de dependencias	219
3.2.4	Perfilar la API: primer escenario tras la refactorización	225
3.3	Perfilar la API: segundo escenario	227
3.3.1	Análisis del consumo de memoria	227
3.3.2	Primera optimización	233
3.3.3	Benchmark de la descompresión del archivo de contrato	237
3.3.4	Segunda optimización	240
3.3.5	Implementar el pooling	244
4.	Conclusión	248

## Capítulo 7

### Más allá del perfilado

1.	Introducción	251
2.	Pistas de mejora restantes	252
2.1	Introducción	252
2.2	Mejorar la experiencia	253
2.3	Tiempo de carga de la aplicación	257
2.4	Procesamiento asíncrono	264
2.5	Marcar los cambios de aplicación	267
2.6	Algunas cuestiones finales	267
2.6.1	Gestión correcta de las trazas	268
2.6.2	Duplicar las consultas SQL	268
2.6.3	Evitar el exceso de arquitectura	270
2.6.4	Paginar los resultados	271
2.6.5	El recolector de basura lleva tiempo	271
2.6.6	Limitar las excepciones	272
2.6.7	Funciones Equals y GetHashCode	274

- 2.6.8 AddRange ..... 275
- 2.6.9 Concatenar cadenas ..... 277
- 3. Tuning ..... 280
  - 3.1 Cuidado ..... 280
  - 3.2 Compilar en release ..... 281
  - 3.3 El cursor de la consistencia ..... 282
    - 3.3.1 BASE en lugar de ACID ..... 282
    - 3.3.2 Un segundo ejemplo ..... 284
    - 3.3.3 Pasar el código de PROFÍ a BASE ..... 285
  - 3.4 Asincronía globalizada ..... 290
  - 3.5 Utilizar referencias débiles ..... 290
  - 3.6 Cuidado con el tuning extremo ..... 291
    - 3.6.1 Límites del tuning ..... 291
    - 3.6.2 Struct en lugar de class ..... 292
    - 3.6.3 Instanciación tardía y suspensión precoz ..... 294
    - 3.6.4 ¿Byte en lugar de int en Enum? ..... 294
- 4. Ir más lejos mediante la rearquitectura ..... 295
  - 4.1 Problemática ..... 295
  - 4.2 Escalabilidad ..... 296
    - 4.2.1 Concepto ..... 296
    - 4.2.2 Modos de escalabilidad ..... 297
    - 4.2.3 Paralelización de los procesamientos ..... 297
    - 4.2.4 Mejores prácticas para la escalabilidad ..... 298
    - 4.2.5 Parallel Linq ..... 299
  - 4.3 Institucionalizar la caché ..... 301
  - 4.4 Pensar Lean/Agile ..... 301
    - 4.4.1 IMDB ..... 302
    - 4.4.2 NoSQL ..... 302
    - 4.4.3 CQRS ..... 302
    - 4.4.4 Prevalencia ..... 303

# 8 \_\_\_\_\_ Escribir código .NET eficaz

Perfilado, benchmarking y buenas prácticas

4.5 Rendimiento de las nuevas arquitecturas . . . . .	304
4.5.1 Problemática . . . . .	304
4.5.2 Scale Out . . . . .	304
4.5.3 Paralelización. . . . .	304
4.5.4 Movilidad. . . . .	305
4.5.5 SOA/EDA/ESB . . . . .	305
4.6 Y para ir aún más lejos . . . . .	307

## Conclusión

1. Todo puede causar problemas . . . . .	309
2. Lista de verificación . . . . .	310
3. Las causas de los errores . . . . .	312
4. Codificar ligero . . . . .	314
5. Conclusión . . . . .	315

Índice . . . . .	317
------------------	-----