DO CONSOLE AO JOGO ONLINE: C# EM 1 ANO

Caderno de Exercícios

73 aulas de prática, do #01 ao #73

C# 14 · .NET 10 · 10 módulos · 88 questões · 2 desafios integradores

Autor Desconhecido · theknowledgeisle.com

Nome do aluno: Início: Término:

Como usar este caderno

Cada módulo reúne um bloco de aulas do curso, com a lista dos artigos que ele cobre. As perguntas conceituais pedem explicações com as suas palavras; os exercícios de código pedem programas: escreva, compile e execute cada um.

Tente sempre resolver sozinho antes de abrir o **Caderno de Respostas**. É no erro que o conhecimento fixa.

Sumário

Módulo	Tema	Aulas	Questões
MÓDULO 1	Fundamentos da linguagem	#01–#08	Q1–Q13
MÓDULO 2	Orientação a objetos	#09–#15	Q14–Q23
MÓDULO 3	Genéricos, delegates e LINQ	#16–#21	Q24–Q32
MÓDULO 4	A plataforma por dentro	#22–#30	Q33–Q44
MÓDULO 5	Testes e bancos de dados	#31–#42	Q45–Q53
MÓDULO 6	ASP.NET Core e serviços	#43–#52	Q54–Q65
MÓDULO 7	A vitrine web e o deploy	#53–#59	Q66–Q72
MÓDULO 8	Aplicativos de desktop e celular	#60–#62	Q73–Q77
MÓDULO 9	Jogos	#63–#69	Q78–Q82
MÓDULO 10	Arquitetura, IA e desempenho	#70–#73	Q83–Q88
DESAFIOS	Desafios integradores	todas	D1–D2

Legenda: nas perguntas, `assim` indica código, nomes de tipos e comandos. As linhas pautadas são para a sua resposta; nos exercícios de código, use também o computador: **todo programa deve ser compilado e executado**.

MÓDULO 1

Fundamentos da linguagem

A história do C#, o .NET por dentro, tipos e conversões, texto e console, decisões e laços, métodos, coleções e exceções.

Artigos 01 a 08

#01 Da Rivalidade com o Java ao Código Aberto: a História do C#
#02 Do Código-Fonte ao Processo: Instalando o .NET e Entendendo o que Acontece ao Rodar
#03 Cada Valor Tem um Tipo: Variáveis, Números e Conversões em C#
#04 Operadores, Texto e Conversa com o Usuário
#05 O Programa Que Escolhe e Repete: if, switch e Laços
#06 Métodos: Dividindo o Problema em Partes Que Conversam
#07 Guardando Muitos Valores: Arrays, Listas e Dicionários
#08 Quando Algo Dá Errado: Exceções em C#

Perguntas conceituais

Q1 **Do código ao processo**

Descreva o caminho de um `Program.cs` até a execução: o que o compilador gera, o que é IL e o que o JIT faz. Por que o mesmo `.dll` roda em Windows, Linux e macOS?

Q2 **Tipos de valor e de referência**

Qual a diferença entre `int` (tipo de valor) e `string` ou uma `class` (tipos de referência) numa atribuição `b = a`? Por que a `string`, sendo de referência, se comporta "como valor"?

Q3 decimal, double e dinheiro

Por que `0.1 + 0.2 == 0.3` é falso com `double` e verdadeiro com `decimal`? Qual usar para preços na livraria do curso, e qual o custo?

Q4 Parse ou TryParse

Qual a diferença entre `int.Parse(texto)` e `int.TryParse(texto, out int n)`? Qual usar para ler o que o usuário digitou, e por quê?

Q5 ref, out e in

Explique a diferença entre os modificadores de parâmetro `ref`, `out` e `in`. Por que o C# exige escrever `ref` também na chamada?

Q6 List, array ou Dictionary

Para cada caso, escolha entre `T[]`, `List<T>` e `Dictionary<TKey, TValue>`: (a) as notas de um aluno numa prova com 10 questões; (b) os itens de um carrinho de compras; (c) buscar um livro pelo ISBN.

Q7 **Quando capturar uma exceção**

"Capture apenas as exceções que você sabe tratar." Explique a regra e por que `catch (Exception) { }` vazio é um dos piores trechos que um programa pode ter.

Exercícios de código

Q8 **Lendo um número com segurança**

Escreva um programa que leia linhas da entrada até receber um inteiro entre 1 e 100, avisando quando o texto não for número e quando estiver fora do intervalo. Use `int.TryParse`. Em seguida, mostre o número, o dobro e se é par.

Q9 Formatando valores em reais

Mostre o preço `1234.5m` formatado: (a) com a cultura `pt-BR` como moeda; (b) com duas casas e separador de milhar em `pt-BR`; (c) na cultura invariante. Depois leia o texto `"1.234,50"` de volta para `decimal` usando `pt-BR`.

Q10 switch moderno

Escreva `static string Classificar(int idade)` usando uma expressão `switch` com padrões relacionais (`< 0`, `< 13`, `< 18`, `< 60`, `_`), devolvendo "inválida", "criança", "adolescente", "adulto" ou "idoso". Depois escreva `static decimal Frete(string uf, decimal total)` com uma expressão `switch` sobre uma tupla: frete grátis acima de R$ 200, R$ 10 para SP e RJ, R$ 25 para as demais UFs.

Q11 Métodos com parâmetros opcionais e out

Escreva `static bool TentarDividir(int a, int b, out int quociente, out int resto)` e `static string Repetir(string texto, int vezes = 2, string separador = " ")`. Chame o segundo com argumentos nomeados, pulando um dos opcionais.

Q12 Estoque com Dictionary

Escreva um programa que mantenha o estoque de uma livraria num `Dictionary<string, int>` (título → quantidade), processe uma lista de movimentações (`"+Dom Casmurro:3"`, `"-Vidas Secas:1"`) e recuse saídas maiores que o estoque. Mostre o estoque final em ordem alfabética.

Q13 **Exceções próprias**

Crie `class EstoqueInsuficienteException : Exception` com as propriedades `Titulo` e `Disponivel`. Escreva `static void Vender(Dictionary<string, int> estoque, string titulo, int qtd)` que a lance quando faltar estoque e `ArgumentOutOfRangeException` para quantidade menor que 1. Trate as duas de forma diferente e use `finally` para registrar o fim de cada tentativa.

MÓDULO 2

Orientação a objetos

Classes e objetos da livraria, propriedades, herança, interfaces, class/struct/record, enums e pattern matching, namespaces, projetos e pacotes NuGet.

Artigos 09 a 15

#09 Classes e Objetos: Dando Forma aos Dados da Livraria
#10 Propriedades e Encapsulamento: Deixar Ler, Controlar a Escrita
#11 Herança: Quando um Tipo É uma Versão Especializada de Outro
#12 Interfaces: Contratos Que Qualquer Tipo Pode Assinar
#13 Classe, Struct ou Record: Identidade, Valor e Igualdade
#14 Enums e Pattern Matching: Nomes para Estados e Perguntas sobre Formas
#15 Do Arquivo Único à Solução: Namespaces, Projetos e Pacotes NuGet

Perguntas conceituais

Q14 Campo ou propriedade

Por que, em C#, os dados públicos de uma classe são expostos como propriedades (`public decimal Preco { get; private set; }`) e não como campos públicos? Cite duas vantagens concretas.

Q15 Herança ou interface

A livraria vende livros físicos e e-books. Quando modelar `LivroFisico` e `Ebook` herdando de uma classe `Produto`, e quando fazer os dois implementarem uma interface `IVendavel`? Por que o C# permite várias interfaces, mas uma só classe base?

Q16 class, struct ou record

Para cada tipo da livraria, escolha `class`, `struct`, `record` ou `record struct`, justificando: (a) `Cliente`, com identidade e dados que mudam; (b) `Isbn`, um código de 13 dígitos; (c) `ItemDoPedido` (título, quantidade e preço unitário), comparado por valor; (d) `Coordenada` com dois `double`, criada aos milhões num mapa.

Q17 Pattern matching

O que este método devolve para um `Ebook` de 30 reais, para um `LivroFisico` de 0,5 kg e para `null`? Por que a ordem dos casos importa?

```
static decimal Frete(Produto? p) => p switch
{
    null => throw new ArgumentNullException(nameof(p)),
    Ebook => 0m,
    LivroFisico { PesoKg: <= 1 } => 10m,
    LivroFisico l => 10m + (decimal)(l.PesoKg - 1) * 5m,
    _ => 15m,
};
```

Q18 **Pacotes e projetos**

Qual a diferença entre uma *solution* (`.sln` ou `.slnx`), um projeto (`.csproj`) e um pacote NuGet? Como a livraria seria dividida em projetos, e o que significa um `<PackageReference Include="Dapper" Version="2.1.66" />`?

Exercícios de código

Q19 **Uma classe com invariante**

Escreva `class Livro` com `Titulo` (obrigatório, não vazio), `Preco` (não negativo, com `private set`) e o método `AplicarDesconto(decimal percentual)`, que aceita de 0 a 50% e arredonda para centavos. Use `ArgumentException` e `ArgumentOutOfRangeException`. Mostre uma criação válida, um desconto e duas tentativas inválidas.

Q20 Herança com métodos virtuais

Escreva `abstract class Produto` com `Titulo`, `PrecoBase` e `public virtual decimal PrecoFinal() => PrecoBase;`, e duas derivadas: `LivroFisico` (acrescenta R$ 5 de embalagem) e `Ebook` (10% mais barato). Some o total de um carrinho com os dois tipos numa `List<Produto>` e mostre por que `ToString()` sobrescrito ajuda a depurar.

Q21 Interfaces e polimorfismo sem herança

Defina `interface IDescontavel { decimal Desconto(decimal total); }` e três políticas: `SemDesconto`, `Percentual(decimal pct)` e `ProgressivoPorQuantidade` (5% a partir de 3 itens, 10% a partir de 5). Escreva um método que escolha a política mais vantajosa para o cliente entre várias.

Q22 Records e igualdade

Escreva `record ItemDoPedido(string Titulo, int Quantidade, decimal PrecoUnitario)` com uma propriedade calculada `Subtotal`. Mostre: a igualdade por valor entre dois itens iguais; o `ToString` gerado; uma cópia com `with` mudando a quantidade; e um `HashSet` que elimina itens repetidos. Compare com uma `class` comum de mesmos campos.

Q23 Enum e estados do pedido

Modele `enum StatusPedido { Novo, Pago, Enviado, Entregue, Cancelado }` e escreva `static bool PodeIr(StatusPedido de, StatusPedido para)` com uma expressão `switch` sobre a tupla: só é permitido Novo → Pago, Pago → Enviado, Enviado → Entregue e cancelar a partir de Novo ou Pago. Teste todas as transições a partir de `Pago`. Mostre também o que `Enum.IsDefined` evita.

MÓDULO 3

Genéricos, delegates e LINQ

Genéricos, delegates, lambdas e eventos, LINQ (filtrar, agrupar, cruzar e a execução adiada), nullable reference types e sequências com yield.

Artigos 16 a 21

#16 Genéricos: Um Código, Muitos Tipos, Nenhuma Conversão
#17 Delegates, Lambdas e Eventos: Passando Comportamento Como Se Fosse Dado
#18 LINQ, Parte I: Perguntas ao Catálogo em Vez de Laços
#19 LINQ, Parte II: Agrupar, Cruzar e Entender Quando a Consulta Executa
#20 Nullable Reference Types: o Fim Anunciado do NullReferenceException
#21 Sequências Sob Demanda: IEnumerable, yield e Coleções Imutáveis

Perguntas conceituais

Q24 **Por que genéricos**

Antes dos genéricos (C# 2), a coleção padrão era o `ArrayList`, que guarda `object`. Cite dois problemas que a `List<int>` resolve em relação a ele.

Q25 **Func, Action e Predicate**

O que representam `Func<int, int, int>`, `Action<string>` e `Func<Livro, bool>`? Escreva uma lambda compatível com cada um.

Q26 Execução adiada

Por que este código imprime 3, e não 2? O que muda com `.ToList()` no fim da primeira linha?

```
var lista = new List<int> { 1, 2, 3, 4 };
var grandes = lista.Where(x => x > 2);
lista.Add(5);
Console.WriteLine(grandes.Count());
```

Q27 Nullable reference types

Com `<Nullable>enable</Nullable>`, qual a diferença entre `string` e `string?`? O que o operador `!` (*null-forgiving*) faz, e por que usá-lo com moderação?

Exercícios de código

Q28 Um repositório genérico

Escreva `interface IEntidade { int Id { get; } }` e `class RepositorioEmMemoria<T> where T : IEntidade` com `Adicionar`, `ObterPorId` (devolve `T?`) e `Todos()` (devolve `IReadOnlyList<T>`). Recuse IDs repetidos. Use-o com `Livro` e com `Cliente`.

Q29 Eventos de estoque

Escreva `class Estoque` com o evento `public event EventHandler<EstoqueBaixoEventArgs>? EstoqueBaixo`, disparado quando uma retirada deixa um título abaixo de 2 unidades. Inscreva dois ouvintes (um que "envia e-mail" e um que registra em log), retire do estoque e mostre a ordem das notificações. Depois cancele uma das inscrições.

Q30 Consultas LINQ no catálogo

Dado um catálogo de livros (título, autor, ano, preço, gênero), escreva consultas LINQ para: (a) os títulos de romance anteriores a 1950, em ordem de ano; (b) o preço médio por gênero; (c) o autor com mais livros; (d) se existe algum livro acima de R$ 100; (e) as páginas de 2 livros, ordenadas por título, mostrando a página 2.

Q31 Juntando pedidos e clientes

Com uma lista de clientes (id, nome, cidade) e uma de pedidos (id, idCliente, valor), use `Join` para listar pedido, cliente e valor; use `GroupJoin` para mostrar o total gasto por cliente, incluindo quem não comprou nada (total 0).

Q32 Sequências com yield

Escreva `static IEnumerable<long> Fibonacci()`, infinita, com `yield return`, e use LINQ para pegar os 10 primeiros, o primeiro acima de 1000 e a soma dos pares menores que 4 milhões. Depois escreva `static IEnumerable<string> LerLinhas(string caminho)` que leia um arquivo de forma preguiçosa e explique por que a sequência infinita não trava o programa.

MÓDULO 4

A plataforma por dentro

Reflexão e atributos, source generators, memória e coletor de lixo, Span<T>, arquivos e JSON, async/await, concorrência e interop com C.

Artigos 22 a 30

#22 Reflexão e Atributos: Quando o Programa Examina a Si Mesmo
#23 Source Generators: Código Que Escreve Código Durante a Compilação
#24 Memória no .NET: Pilha, Heap, Coletor de Lixo e IDisposable
#25 Span<T> e Memory<T>: Olhar Sem Copiar, Processar Sem Alocar
#26 Arquivos, Streams e JSON: Dados Que Sobrevivem ao Programa
#27 async e await: Esperar Sem Bloquear
#28 Assincronia Avançada: Muitas Tarefas, Cancelamento e Fluxos Assíncronos
#29 Concorrência: Threads, Travas e Dados Compartilhados Sem Corrupção
#30 Interop com C: Chamando Código Nativo com P/Invoke e unsafe

Perguntas conceituais

Q33 **Reflexão ou source generator**

Um serializador pode descobrir as propriedades de um tipo por **reflexão**, em execução, ou gerá-las com um **source generator**, na compilação. Compare as duas abordagens em desempenho, em compatibilidade com Native AOT e em facilidade de depuração.

Q34 **Gerações do coletor de lixo**

Explique as gerações 0, 1 e 2 do coletor de lixo do .NET e a hipótese em que elas se apoiam. O que é o LOH?

Q35 IDisposable e using

Se o .NET tem coletor de lixo, por que existe `IDisposable`? O que `using var arquivo = File.OpenRead(...)` garante, e o que é um *finalizer*?

Q36 async não é thread

"Um método `async` roda em outra thread." Por que a frase está errada? O que acontece de fato num `await File.ReadAllTextAsync(...)` quando o arquivo ainda não foi lido?

Q37 Task.Run e async void

Por que `Task.Run` não deve envolver chamadas de E/S num servidor ASP.NET Core, e por que `async void` só é aceitável em *handlers* de evento?

Q38 lock e o que ele protege

O que `lock (_trava) { ... }` garante, e por que a trava deve ser um objeto privado, e não `this` ou um `string`? O que o .NET 9 trouxe para isso?

Exercícios de código

Q39 Atributos e reflexão

Crie o atributo `[Obrigatorio]` e um validador genérico que percorra, por reflexão, as propriedades `string` de qualquer objeto e liste as marcadas com o atributo que estejam vazias. Teste com uma classe `CadastroCliente`.

Q40 Span sem alocar

Escreva `static int SomarCsv(ReadOnlySpan<char> linha)` que some os inteiros de uma linha como `"10;20;x;30"`, ignorando os campos inválidos, sem criar nenhuma `string` (nada de `Split`). Compare a quantidade de bytes alocados com a versão que usa `Split` e `int.Parse`, medindo com `GC.GetAllocatedBytesForCurrentThread()`.

Q41 Gravando e lendo JSON

Serialize uma lista de `record Livro(string Titulo, decimal Preco, DateOnly Lancamento)` para um arquivo JSON com nomes em *camelCase* e indentação, leia o arquivo de volta e confira que as listas são iguais. Mostre o JSON gravado.

Q42 Várias tarefas com cancelamento

Simule a consulta de preço de um livro em três fornecedores com tempos de resposta diferentes (`Task.Delay` de 300, 100 e 2000 ms). Use `Task.WhenAll` para esperar os que respondem em até 500 ms, cancelando o lento com um `CancellationTokenSource` com tempo limite, e mostre o menor preço entre os que responderam.

Q43 Contando em paralelo sem corrupção

Conte as palavras de 4 textos em paralelo com `Parallel.ForEach`, acumulando num `Dictionary<string, int>` comum protegido por `lock` e, em outra versão, num `ConcurrentDictionary<string, int>` com `AddOrUpdate`. Confira que os dois dão o mesmo total.

Q44 Chamando C com P/Invoke

Use `[LibraryImport]` para chamar a função `strlen` da biblioteca C do Linux (`libc`) e a função `abs`. Compare com `string.Length` para um texto com acentos e explique a diferença.

MÓDULO 5

Testes e bancos de dados

xUnit, SQL com SQLite, ADO.NET, Dapper, Entity Framework Core, PostgreSQL, MySQL, MongoDB e um repositório independente do banco.

Artigos 31 a 42

#31 Testes Automatizados com xUnit: Mudar Sem Medo
#32 SQL Essencial com SQLite: Perguntas a um Banco de Dados
#33 ADO.NET: Conexões, Comandos Parametrizados e o Perigo do SQL Injection
#34 Dapper: SQL Seu, Sem a Cerimônia
#35 Entity Framework Core: Objetos no Banco Sem Escrever SQL
#36 EF Core: Relacionamentos e Migrations
#37 EF Core com Desempenho: N+1, AsNoTracking e o SQL por Trás de Cada Consulta
#38 PostgreSQL: o Banco de Dados Servidor, com Npgsql, JSONB e Arrays
#39 MySQL: MySqlConnector, o Provider do EF e as Diferenças Que Importam
#40 MongoDB: o Modelo de Documentos e o Driver Oficial
#41 MongoDB Avançado: Agregações, Índices e a Escolha Entre Documento e Relacional
#42 Um Repositório, Três Bancos: Injeção de Dependência e a Livraria Independente de Infraestrutura

Perguntas conceituais

Q45 Fact, Theory e o padrão AAA

No xUnit, qual a diferença entre `[Fact]` e `[Theory]` com `[InlineData]`? O que é o padrão *Arrange, Act, Assert*?

Q46 SQL injection

Mostre por que o código abaixo é vulnerável, o que um atacante digitaria e como o ADO.NET resolve o problema.

```
var cmd = conexao.CreateCommand();
cmd.CommandText = $"SELECT * FROM clientes WHERE email = '{email}'";
```

Q47 **ADO.NET, Dapper ou EF Core**

Compare ADO.NET puro, Dapper e EF Core quanto ao que você escreve, ao controle sobre o SQL e ao que cada um faz por você. Quando escolher cada um?

Q48 **O problema N+1**

O que é o problema N+1 no EF Core? Mostre o código que o causa e duas formas de corrigir.

Q49 **Documento ou relacional**

A livraria guarda pedidos com itens. Quando modelar no MongoDB, com os itens **embutidos** no documento do pedido, é melhor que no PostgreSQL, com tabelas `pedidos` e `itens`? E quando é pior?

Exercícios de código

Q50 **Testes com xUnit**

Escreva, como projeto de testes xUnit, os testes da regra de frete do módulo 1 (`Frete(uf, total)`): uma `[Theory]` cobrindo SP, RJ, outra UF e o limite de R$ 200 (exatamente 200 e 200,01), e um `[Fact]` que verifique que uma UF em minúsculas é tratada como as outras UFs (documentando o comportamento atual).

Q51 ADO.NET com SQLite

Com o pacote `Microsoft.Data.Sqlite`, crie um banco em memória, a tabela `livros(id, titulo, preco)`, insira três livros com um comando parametrizado dentro de uma transação e liste os que custam menos de R$ 50, lendo com `SqliteDataReader`.

Q52 EF Core em memória com relacionamentos

Com o EF Core e o provedor SQLite em memória, modele `Autor` (um) e `Livro` (muitos), crie o banco com `EnsureCreated`, grave dois autores com livros e liste cada autor com a quantidade de livros e o mais caro, usando uma projeção. Diga como ver o SQL que o EF gerou.

Q53 Um repositório, vários bancos

Defina `interface ILivroRepositorio { Task<Livro?> ObterAsync(int id); Task AdicionarAsync(Livro l); }`, uma implementação em memória e um `CatalogoService` que dependa só da interface. Registre tudo num `ServiceCollection` (injeção de dependência) e mostre que trocar a implementação é uma linha.

MÓDULO 6

ASP.NET Core e serviços

Hosting e pipeline de middleware, a API da livraria, validação e ProblemDetails, JWT, configuração e logging, cache, SignalR, workers e mensageria, gRPC e produção.

Artigos 43 a 52

#43 ASP.NET Core por dentro: hosting, pipeline de middleware e Minimal APIs
#44 A API da Livraria: recursos, verbos, DTOs e concorrência sobre HTTP
#45 Validação, ProblemDetails e OpenAPI: uma API que explica a si mesma
#46 Autenticação e autorização com JWT: quem é você e o que pode fazer
#47 Configuração, logging e testes de integração: a API pronta para ser operada
#48 Cache com Redis, HybridCache e output caching: responder sem ir ao banco
#49 Tempo real com SignalR: quando o servidor toma a iniciativa
#50 Worker Services e mensageria com RabbitMQ: trabalho fora da requisição
#51 gRPC: contratos binários e streaming entre serviços
#52 Produção: CORS, rate limiting, versionamento e OpenTelemetry

Perguntas conceituais

Q54 A ordem dos middlewares

O pipeline do ASP.NET Core é uma sequência de *middlewares*. Por que `app.UseAuthentication()` precisa vir antes de `app.UseAuthorization()`, e o que acontece se um *middleware* não chamar `next()`?

Q55 Entidade ou DTO

Por que a API da livraria não deve receber nem devolver diretamente a entidade `Livro` do EF Core? O que é um DTO, e que ataque o *overposting* permite?

Q56 **Concorrência otimista com ETag**

Dois atendentes abrem o mesmo livro, e cada um altera o preço e salva. Como `ETag` e `If-Match` evitam que a segunda gravação apague a primeira sem ninguém perceber? Que código HTTP a API devolve?

Q57 **O que um JWT protege**

Um JWT tem três partes separadas por ponto. O que cada uma contém, e por que é um erro guardar a senha ou o CPF do usuário dentro dele?

Q58 **De onde vem a configuração**

No ASP.NET Core, a mesma chave pode vir do `appsettings.json`, do `appsettings.Production.json`, de variáveis de ambiente e da linha de comando. Qual vence, e onde deve ficar a *connection string* de produção?

Q59 **Cache: o problema difícil**

"Há duas coisas difíceis em computação: invalidar cache e dar nomes às coisas." Ao guardar o catálogo da livraria no Redis, que estratégias existem para que um preço alterado não continue aparecendo velho, e o que é uma *cache stampede*?

Q60 **Entrega pelo menos uma vez**

O RabbitMQ garante entrega "pelo menos uma vez" quando o consumidor só confirma (`ack`) depois de processar. Por que isso obriga o consumidor que envia o e-mail de confirmação de pedido a ser **idempotente**, e como fazê-lo?

Q61 **gRPC ou REST**

Para a comunicação entre o serviço de pedidos e o de estoque, por que gRPC pode ser melhor que uma API REST com JSON? Por que ele não costuma ser usado diretamente pelo navegador?

Exercícios de código

Q62 O pipeline em ação

Escreva uma aplicação ASP.NET Core mínima com dois *middlewares* próprios: um que mede o tempo e acrescenta o cabeçalho `X-Tempo-Ms`, e outro que recusa com 403 qualquer requisição para `/admin` sem o cabeçalho `X-Chave: segredo`. Suba o servidor numa porta local, faça três requisições com `HttpClient` dentro do próprio programa e mostre a ordem em que os *middlewares* rodam.

Q63 Uma API de livros com DTOs e ProblemDetails

Escreva os *endpoints* `GET /livros/{id}` (200 ou 404), `POST /livros` (201 com o cabeçalho `Location`, ou 400 com `ValidationProblem` quando o título está vazio ou o preço não é positivo) e `PUT /livros/{id}` com `If-Match` (412 se a versão não bater). Use DTOs de entrada e saída e um dicionário como armazenamento. Teste com `HttpClient` no próprio programa.

Q64 **Limitando requisições**

Use o *rate limiter* embutido do ASP.NET Core para permitir no máximo 3 requisições a cada 10 segundos em `/busca`, respondendo 429 às excedentes. Faça 5 requisições seguidas e mostre os códigos.

Q65 **Um worker com fila em memória**

Sem RabbitMQ, simule o padrão de mensageria com `System.Threading.Channels`: um `Channel<T>` limitado a 2 itens recebe eventos `PedidoCriado`. Mostre que a fila recusa um terceiro evento enquanto ninguém consome (`TryWrite`), depois inicie um consumidor que processa cada evento de forma idempotente, ignorando um evento entregue duas vezes, e publique o restante com `WriteAsync`, que espera quando a fila enche.

MÓDULO 7

A vitrine web e o deploy

Razor Pages, formulários e validação, Blazor e os modos de renderização, o site consumindo a API, login e upload, e o deploy com contêineres, nginx e Aspire.

Artigos 53 a 59

#53 Razor Pages: HTML montado no servidor para a vitrine da Livraria
#54 Formulários e validação: o site recebe dados
#55 Blazor: componentes, parâmetros, eventos e ciclo de vida
#56 Blazor: modos de renderização, WebAssembly e estado
#57 O site consumindo a API: clientes resilientes, cache condicional e um BFF com YARP
#58 Login, área administrativa, upload de capas e notificações em tempo real
#59 Deploy: contêineres, nginx, Linux e a orquestração com Aspire

Perguntas conceituais

Q66 Post, Redirect, Get

Depois de processar o formulário de "finalizar pedido", por que a página deve responder com um redirecionamento (303 ou 302) para uma página de confirmação, e não renderizar a confirmação direto?

Q67 O token antiforgery

O que o *token antiforgery* das Razor Pages protege, e por que ele não é necessário numa API que só aceita `Authorization: Bearer`?

Q68 Modos de renderização do Blazor

Compare os modos *Static SSR*, *Interactive Server* e *Interactive WebAssembly* do Blazor quanto a onde o código roda, latência das interações e custo para o servidor. Qual usar na página de um livro, no carrinho e no painel administrativo?

Q69 Cache condicional

O site consulta `GET /livros/1` na API a cada visita. Como `ETag` e `If-None-Match` economizam banda, e que código a API devolve quando nada mudou?

Exercícios de código

Q70 Validando um upload de capa

Escreva `static string? ValidarCapa(string nomeArquivo, long tamanho, ReadOnlySpan<byte> inicio)` que devolva uma mensagem de erro (ou `null` se estiver tudo certo) para o upload da capa de um livro: aceite só JPEG e PNG, conferindo a extensão **e** os primeiros bytes do arquivo (a "assinatura"), com até 2 MB. Teste com um PNG válido, um "PNG" que na verdade é um executável, uma capa grande demais e um nome sem extensão.

Q71 Repetição com espera exponencial

O site chama a API, que às vezes falha por instantes. Escreva `static async Task<T> ComRepeticao<T>(Func<Task<T>> acao, int tentativas, TimeSpan esperaInicial)` que repita a ação em caso de `HttpRequestException`, dobrando a espera a cada falha, e desista depois do número de tentativas, relançando o último erro. Simule uma API que falha duas vezes e depois responde.

Q72 Um Dockerfile para a API

Escreva um `Dockerfile` em múltiplos estágios para a API da livraria: um estágio com o SDK que restaura e publica, e um estágio final só com o *runtime* do ASP.NET, rodando com um usuário sem privilégios. Explique por que copiar o `.csproj` antes do restante do código acelera os *builds*.

MÓDULO 8

Aplicativos de desktop e celular

WinForms, WPF, MAUI e Avalonia; o caixa da livraria com MVVM e vendas offline; o aplicativo do cliente em .NET MAUI.

Artigos 60 a 62

#60 Interfaces no .NET: WinForms, WPF, MAUI, Avalonia e o que escolher
#61 O caixa da Livraria em Avalonia: MVVM, navegação, leitor de código de barras e vendas offline
#62 O aplicativo do cliente em .NET MAUI: Shell, listas, login no celular e acompanhamento de pedidos

Perguntas conceituais

Q73 Qual framework de interface

O caixa da livraria precisa rodar em Windows e em Linux; o aplicativo do cliente, em Android e iOS. Por que o curso escolheu Avalonia para o primeiro e .NET MAUI para o segundo, e onde WinForms e WPF ainda fazem sentido?

Q74 A regra da thread de interface

Por que um clique que faz `var dados = http.GetStringAsync(url).Result;` congela a janela, e por que alterar um controle a partir de uma thread de segundo plano lança exceção?

Q75 Por que MVVM

O que o padrão MVVM separa, e qual a vantagem concreta de ter a lógica do caixa num *ViewModel* em vez de nos eventos de clique da janela?

Exercícios de código

Q76 Um ViewModel testável

Escreva `class CaixaViewModel : INotifyPropertyChanged` com uma lista de itens, a propriedade calculada `Total`, um `CommandoSimples` (implementação mínima de `ICommand`) para finalizar a venda, habilitado só com itens no carrinho, e o método `Adicionar(string titulo, decimal preco)`. Sem interface gráfica, mostre as notificações disparadas e o estado do comando.

Q77 Vendas offline

O caixa precisa continuar vendendo quando a internet cai. Escreva uma `FilaOffline` que guarde as vendas pendentes, e um método `SincronizarAsync(Func<Venda, Task<bool>> enviar)` que tente enviar cada uma **em ordem**, parando na primeira falha (para não enviar a venda 3 antes da 2), e removendo da fila só as que foram aceitas. Simule uma queda no meio da sincronização.

MÓDULO 9

Jogos

Unity, Godot e MonoGame; o game loop do zero; nós, cenas e sinais; física e colisões; arquitetura sem alocar no loop; áudio, interface e save game.

Artigos 63 a 69

#63 C# nos jogos: Unity, Godot, MonoGame e o game loop
#64 Um game loop do zero com MonoGame: janela, sprites, entrada e os primeiros livros caindo
#65 Godot com C#: nós, cenas e sinais
#66 Input, física e colisões: fazendo o jogo parecer justo
#67 Arquitetura de jogo e por que não alocar dentro do loop
#68 Áudio, interface e save game: o que faz um jogo parecer pronto
#69 Projeto: a Biblioteca em Apuros completa, do documento de design ao export

Perguntas conceituais

Q78 **Unity, Godot ou MonoGame**

Compare Unity, Godot e MonoGame quanto ao que cada um oferece pronto e ao modelo de licença. Por que o curso começa o game loop com MonoGame e depois passa para Godot?

Q79 **Por que não alocar no loop**

Um jogo roda a 60 quadros por segundo e cria, a cada quadro, uma `List<Inimigo>` temporária e algumas strings para o placar. Por que isso causa "engasgos" periódicos, e que técnicas evitam o problema?

Q80 Sinais do Godot

No Godot, um nó `Moeda` emite o sinal `Coletada` quando o jogador a toca, e o nó `Placar` reage a ele. Que vantagem isso tem sobre a moeda chamar `GetNode<Placar>("/root/Fase/Placar").Somar(10)` diretamente?

Exercícios de código

Q81 Um pool de projéteis

Escreva `class PoolDeProjeteis` que crie 3 projéteis de antemão, entregue um inativo em `Disparar()` (ou `null` se todos estiverem em uso) e os devolva ao *pool* quando saem da tela. Simule 10 quadros disparando a cada quadro e mostre que o número de objetos criados não passa de 3. Compare as alocações com uma versão que cria um projétil novo a cada disparo.

Q82 Save game com versão

Escreva `record SaveGame(int Versao, int Fase, int Pontos, List<string> Itens)` e funções para salvar em JSON e carregar, migrando saves antigos: a versão 1 não tinha `Itens`. Carregue um save da versão 1 escrito à mão, migre para a versão 2 e recuse um arquivo corrompido sem derrubar o jogo.

MÓDULO 10

Arquitetura, IA e desempenho

Clean Architecture e DDD com moderação, C# e IA com Microsoft.Extensions.AI e busca semântica, BenchmarkDotNet, Native AOT e ferramentas de linha de comando.

Artigos 70 a 73

#70 Arquitetura: Clean Architecture, DDD com moderação e os padrões que já usamos
#71 C# e IA: Microsoft.Extensions.AI, busca semântica e um assistente com ferramentas
#72 Performance e distribuição: BenchmarkDotNet, Native AOT e ferramentas de linha de comando
#73 Capstone: o jogo, a API, o banco e o site num só fluxo, e a retrospectiva do curso

Perguntas conceituais

Q83 A regra da dependência

Na Clean Architecture, as camadas são Domínio, Aplicação, Infraestrutura e Apresentação. Para que lado apontam as referências entre os projetos, e por que o `Livraria.Dominio` não deve referenciar o pacote do EF Core?

Q84 Ferramentas para um modelo de IA

Num assistente da livraria feito com `Microsoft.Extensions.AI`, o modelo pode chamar a função `BuscarLivros(string termo)`. Quem executa a função de fato, e que cuidados de segurança isso exige?

Q85 Medir antes de otimizar

Por que um laço com `Stopwatch` em volta de uma chamada é uma forma pouco confiável de comparar duas implementações, e o que o BenchmarkDotNet faz de diferente?

Q86 Native AOT: o que se ganha e o que se perde

O que muda ao publicar uma ferramenta de linha de comando com `PublishAot=true`? Cite dois ganhos e duas restrições.

Exercícios de código

Q87 Um objeto de valor Dinheiro

No espírito do DDD, escreva `readonly record struct Dinheiro(decimal Valor, string Moeda)` que recuse valores com mais de duas casas, impeça somar moedas diferentes (lançando `InvalidOperationException`) e ofereça `Dividir(int partes)`, que reparte sem perder centavos (o resto vai para as primeiras partes). Divida R$ 100,00 em 3.

Q88 Busca semântica com vetores

Uma busca semântica compara *embeddings*: vetores que um modelo de IA gera para cada texto, próximos quando os significados são próximos. Sem chamar modelo nenhum, use vetores de 4 dimensões já prontos para três sinopses e para a pergunta "livro sobre a seca no sertão", calcule a **similaridade de cosseno** com `TensorPrimitives.CosineSimilarity` e ordene os resultados.

PARA IR ALÉM

Desafios integradores

Projetos maiores, que juntam vários módulos. Não há uma única resposta certa: o gabarito mostra um caminho.

D1

Uma calculadora de terminal

Escreva uma calculadora de terminal com sete operações: soma, subtração, multiplicação, divisão, potência (expoente inteiro), porcentagem (`15 % 200` é 15% de 200) e raiz quadrada (`raiz 2`). Ela deve: (a) aceitar vírgula ou ponto como separador decimal; (b) guardar o último resultado em `ans` , utilizável na conta seguinte; (c) recusar divisão por zero, raiz de negativo, expoente fracionário, resultado grande demais e texto que não é número com uma mensagem clara, **sem encerrar o programa**; (d) ter os comandos `hist` (histórico) e `sair` . Separe o núcleo (uma classe `Calculadora` sem `Console`) da interface, e use `decimal` para que `0,1 + 0,2` dê exatamente `0,3` .

Sua solução (arquitetura, decisões e código principal):

D2

A livraria de ponta a ponta

Junte as peças do curso numa versão mínima da livraria: uma API (Minimal API) com `GET /livros`, `GET /livros/{id}` e `POST /pedidos`, com EF Core e SQLite; um *worker* que consome os pedidos criados de um `Channel` e "envia" o e-mail de confirmação; e um cliente de terminal que lista o catálogo e faz um pedido pela API. Inclua testes de integração da API com `WebApplicationFactory`.

Sua solução (arquitetura, decisões e código principal):

Bom estudo e bons exercícios!

Do Console ao Jogo Online: C# em 1 Ano · theknowledgeisle.com