Advanced .NET Developer Interview Questions & Answers (Senior Level)

1: Explain how you would design a scalable microservice architecture in .NET using Domain-Driven Design (DDD).
A: Use bounded contexts, aggregates, repositories, and domain events. Apply CQRS using MediatR to separate reads and writes. Each bounded context should own its own database for true decoupling.

2: What are the performance trade-offs between using IQueryable, IEnumerable, and AsEnumerable() in LINQ to Entities?
A: IQueryable executes on the database (SQL translation). IEnumerable executes in-memory. AsEnumerable forces client-side evaluation. Use IQueryable for filtering before loading data.

3: How do you handle distributed transactions across multiple microservices?
A: Implement Saga or Outbox Pattern with message brokers (Kafka, RabbitMQ). Ensure eventual consistency and idempotent consumers.

4: Describe how you would implement a custom middleware for request throttling or rate-limiting.
A: Use in-memory or distributed token bucket algorithms. Maintain state in Redis for multi-instance apps. Register middleware early in the request pipeline.

5: How would you diagnose and fix a memory leak in a .NET Core production system?
A: Use tools like dotnet-counters, dotnet-gcdump, PerfView. Look for unbounded collections, event handlers not unsubscribed, or unmanaged resources not disposed.

6: Explain the difference between IHostedService, BackgroundService, and WorkerService. A: IHostedService is a contract, BackgroundService provides an abstract base class, WorkerService is a project template built on BackgroundService.

7: How do you implement resilient HTTP communication between services?
A: Use Polly library for retry, circuit breaker, and fallback policies with exponential backoff. Configure via HttpClientFactory.

8: What is the difference between Span, Memory, and ReadOnlySpan?
A: Span is stack-only and high-performance for slicing arrays. Memory works on the heap and supports async. ReadOnlySpan prevents modification.

9: How would you secure a .NET API hosted in Azure with managed identities and Key Vault?
A: Use Azure AD for authentication. Retrieve secrets via managed identity, eliminating stored credentials. Assign Key Vault access policies per identity.

10: How do you handle versioning in REST APIs and gRPC services?
A: Use URI versioning (v1/v2), header-based versioning, or media-type versioning. For gRPC, manage versions via separate proto files.

11: Explain the internal flow of dependency injection in ASP.NET Core.
A: ServiceCollection builds a ServiceProvider. It resolves dependencies based on lifetime (Transient, Scoped, Singleton). Uses constructor injection for classes.

12: What are the differences between AOT (Ahead-of-Time) compilation and JIT in .NET 8?
A: AOT compiles IL to native at build time improving startup but increasing binary size. JIT compiles at runtime allowing optimizations per environment.

13: How would you design a multitenant SaaS application using .NET 7?

A: Use tenant resolution middleware. Choose per-tenant DB or shared DB with filters. Isolate tenant data using EF Core global query filters.

14: Describe how you would implement event sourcing in .NET.
A: Store state changes as events in an append-only log. Replay events to rebuild state. Use CQRS and versioned event stores for consistency.

15: How do you design a thread-safe caching system in .NET Core for a high-traffic API?
A: Use IMemoryCache for in-memory caching or IDistributedCache for Redis.Ensure thread safety by using GetOrCreate pattern with locking (e.g., SemaphoreSlim). Leverage cache invalidation strategies (sliding/absolute expiration).

16: Explain how you would handle high CPU usage in an ASP.NET Core production system.
A: Use dotnet-trace and PerfView to analyze CPU hotspots.
Profile async tasks for blocking calls. Optimize LINQ and serialization overheads. If thread pool starvation occurs, tune ThreadPool.SetMinThreads.

17: What are the differences between synchronization contexts and task schedulers?
A: SynchronizationContext controls where continuations run (UI thread, ASP.NET request thread).
TaskScheduler schedules tasks for execution (default, custom, or concurrent). Custom schedulers can prioritize or throttle task execution.

18: How does the IAsyncEnumerable<T> interface differ from returning Task<IEnumerable<T>>?
A: IAsyncEnumerable<T> streams data asynchronously using await foreach.
It reduces memory usage by yielding items one-by-one instead of buffering the entire collection.

19: How would you manage database migrations across multiple microservices using EF Core?
A: Maintain independent migration history per bounded context. Use Flyway or Liquibase for centralized schema version control. Apply migrations via CI/CD during deployment phases.

20: What are the best practices for minimizing GC (Garbage Collection) overhead in .NET?
A: Reuse large objects (e.g., via ArrayPool<T>). Avoid boxing/unboxing.
Minimize allocations in tight loops.
Use Span<T> and structs instead of reference-heavy objects.

21: How do you handle async deadlocks in ASP.NET Core?
A: Avoid .Result or .Wait() on async calls. Use ConfigureAwait(false) in library code.
Understand synchronization context differences between console and ASP.NET environments.

22: How does ASP.NET Core handle HTTP request pipelines internally?
A: It uses middleware registered in Startup.Configure.
Each middleware can process, modify, or short-circuit the pipeline.
IApplicationBuilder chains delegates using the decorator pattern.

23: What is the difference between a transient fault and a permanent fault in distributed systems?
A: Transient faults are temporary (network blips, DB timeouts).
Permanent faults require manual intervention (config errors, authentication issues). Handle transients with retry policies; detect permanents via circuit breakers.

24: How do you implement global exception handling and structured logging in .NET?
A: Use UseExceptionHandler() middleware for global errors. Log with Serilog or NLog using structured logs (JSON). Attach correlation IDs for distributed tracing.

25: What are the advantages of using Record types in C# 9+?
A: Records are immutable reference types with built-in value equality. Ideal for DTOs and data models in CQRS.
Support non-destructive mutation via with expressions.

26: How does ASP.NET Core Dependency Injection handle circular dependencies?
Circular dependencies cause runtime InvalidOperationException.
Avoid them by introducing service boundaries or using factory delegates to lazily resolve dependencies.

27: How would you integrate OpenTelemetry in a .NET microservice?
A: Install OpenTelemetry.Exporter.Console or OpenTelemetry.Exporter.Zipkin.
Use AddOpenTelemetryTracing() and instrument HTTP clients, EF Core, and custom spans.
Visualize traces in Jaeger or Application Insights.

28: How do you optimize EF Core queries for performance?
A: Use compiled queries (EF.CompileQuery). Disable change tracking for read-heavy queries.
Load related data with Select projections instead of Include.

29: Use Azure App Configuration or Consul.
A: Each service reads environment-based configurations via Feature Flags or Key-Per- Environment patterns.
Cache configuration locally with refresh intervals.