Here are 25 frequently asked interview questions with detailed answers for a candidate with 8+ years of experience in .NET:

**1. What are the main features introduced in .NET 6?**

**Answer**: .NET 6 includes unified development experience, improved performance, hot reload, new APIs, cross-platform support, minimal APIs for creating HTTP services with minimal code, Blazor improvements, and long-term support (LTS).

**2.Explain the difference between .NET Core and .NET Framework.**

**Answer**: .NET Core is a cross-platform, open-source framework for modern application development, while .NET Framework is Windows-only and primarily used for legacy applications. .NET Core offers better performance, flexibility, and container support.

**3. How does dependency injection work in .NET Core?**

**Answer**: Dependency injection in .NET Core is built-in and configured through the `Startup` class. Services are registered in the `ConfigureServices` method using methods like `AddSingleton`, `AddScoped`, and `AddTransient`. These services are then injected into constructors of classes that need them.

**4. What are minimal APIs in .NET 6, and why are they useful?**

**Answer**: Minimal APIs are a simplified way to create HTTP APIs with less code. They use minimal hosting APIs and a more functional approach to defining routes and handling requests, which reduces boilerplate and enhances performance.

**5. Can you explain the middleware pipeline in ASP.NET Core?**

**Answer**: The middleware pipeline is a sequence of middleware components that HTTP requests pass through. Each middleware can handle the request and/or pass it to the next component. It’s configured in the `Startup.Configure` method using `app.Use...` methods.

**6. What is the purpose of the `IHostedService` interface in .NET?**

**Answer**: `IHostedService` is used to create background services in .NET applications. It provides `StartAsync` and `StopAsync` methods for handling long-running tasks or background processing.

**7. How do you manage configuration in ASP.NET Core applications?**

**Answer**: Configuration is managed using the `IConfiguration` interface and can be sourced from various providers like JSON files, environment variables, command-line arguments, and more. Configuration is typically set up in the `Program.cs` or `Startup.cs` file.

**8. Describe the role of `IActionResult` in ASP.NET Core MVC.**

**Answer**: `IActionResult` is an interface representing a result of an action method. It allows for a variety of return types like `ViewResult`, `JsonResult`, `RedirectResult`, etc., providing flexibility in what the controller can return.

**9. What are the new C# 10 features that are relevant to .NET 6?**

**Answer**: C# 10 introduces features like global using directives, file-scoped namespaces, record structs, interpolated string handlers, and improved lambda expressions. These features enhance code readability, conciseness, and performance.

**10. How do you handle exceptions in ASP.NET Core?**

**Answer**: Exception handling in ASP.NET Core is typically done using middleware such as `UseExceptionHandler` for a global approach and `UseDeveloperExceptionPage` for detailed error information during development. Custom middleware can also be used for more granular control.

**11. Explain Blazor and its use cases.**

**Answer**: Blazor is a framework for building interactive web UIs using C# instead of JavaScript. It comes in two flavors: Blazor Server, which runs on the server and Blazor WebAssembly, which runs client-side in the browser. Blazor is useful for developers who prefer C# and .NET over JavaScript and want to leverage their skills for web development.

**12. What is Entity Framework Core, and how does it differ from Entity Framework 6?**

**Answer**: Entity Framework Core (EF Core) is a lightweight, extensible, open-source version of Entity Framework, designed to be cross-platform. It differs from EF6 by providing better performance, LINQ support, and the ability to work with non-relational databases.

**13. How do you perform database migrations in EF Core?**

**Answer**: Database migrations in EF Core are managed using the `Add-Migration`, `Update-Database`, and `Remove-Migration` commands in the Package Manager Console or `dotnet ef` CLI commands. Migrations allow you to incrementally update the database schema.

**14. What are gRPC services in .NET, and why are they beneficial?**

**Answer**: gRPC is a high-performance, language-agnostic RPC framework. In .NET, it provides efficient communication between microservices with benefits like strong typing, contract-first development, and low latency.

**15. Can you explain the purpose of the `HttpClientFactory` in .NET Core?**

- \*\*Answer:\*\* `HttpClientFactory` is used to create and manage `HttpClient` instances in a centralized manner. It helps avoid common pitfalls like socket exhaustion by handling the lifecycle of `HttpClient` instances properly and supports features like logging, retries, and resilience.

**### 16. \*\*What is the Kestrel web server, and when should you use it?\*\***

- \*\*Answer:\*\* Kestrel is a cross-platform web server for ASP.NET Core. It’s lightweight, high-performance, and used by default in ASP.NET Core applications. Kestrel can be used directly for web applications or as an edge server with a reverse proxy like IIS or Nginx.

**### 17. \*\*How do you secure an ASP.NET Core application?\*\***

- \*\*Answer:\*\* Security in ASP.NET Core can be achieved using authentication and authorization mechanisms, such as Identity, JWT tokens, OAuth2, and OpenID Connect. HTTPS, data protection, and CSP headers are also used to enhance security.

**### 18. \*\*What is the difference between `AddSingleton`, `AddScoped`, and `AddTransient` in** dependency injection?\*\*

- \*\*Answer:\*\* `AddSingleton` creates a single instance for the entire application lifecycle, `AddScoped` creates a new instance per request, and `AddTransient` creates a new instance each time it’s requested.

**### 19. \*\*How do you implement logging in .NET Core?\*\***

- \*\*Answer:\*\* Logging is implemented using the built-in logging framework provided by .NET Core. It supports various providers like Console, Debug, EventSource, and third-party providers like Serilog, NLog, and log4net. Logging is configured in the `Program.cs` or `Startup.cs` file.

**### 20. \*\*What are async and await in C#?\*\***

- \*\*Answer:\*\* `async` and `await` are used to write asynchronous code in C#. `async` is a modifier indicating that a method contains asynchronous operations, and `await` is used to pause the execution until the awaited task completes. This helps in improving application responsiveness.

**### 21. \*\*Explain the concept of microservices and how .NET supports them.\*\***

- \*\*Answer:\*\* Microservices architecture involves breaking down an application into smaller, independently deployable services. .NET supports microservices through ASP.NET Core for building lightweight services, gRPC for efficient communication, Docker for containerization, and tools like Kubernetes for orchestration.

**### 22. \*\*What are the advantages of using Docker with .NET applications?\*\***

- \*\*Answer:\*\* Docker provides consistent environments, easier deployment, scalability, and isolation. It allows .NET applications to run consistently across different environments and simplifies the management of dependencies and configurations.

**### 23. \*\*What is SignalR, and how is it used in .NET?\*\***

- \*\*Answer:\*\* SignalR is a library for adding real-time web functionality to applications. It uses WebSockets and other technologies to provide bi-directional communication between the server and client, enabling real-time features like chat applications, live updates, and notifications.

**### 24. \*\*How do you handle performance tuning in .NET applications?\*\***

- \*\*Answer:\*\* Performance tuning involves profiling and monitoring to identify bottlenecks, optimizing code, using efficient data structures, caching, optimizing database queries, and using asynchronous programming. Tools like dotTrace, BenchmarkDotNet, and built-in .NET Core diagnostics can be used.

**### 25. \*\*What is a `CancellationToken` in .NET, and how is it used?\*\***

- \*\*Answer:\*\* `CancellationToken` is used to signal cancellation of asynchronous operations. It is passed to async methods and checked periodically to see if a cancellation request has been made, allowing tasks to be cancelled gracefully.

These questions and answers should help you practice and prepare for interviews, giving you a solid understanding of key concepts and latest features in .NET.