1. Interview questions for a .NET Core developer with 10 years of experience would likely delve into advanced topics and require in-depth knowledge. Here are some questions you could consider:

2. **Explain the architecture of a .NET Core application**: Ask the candidate to explain the architecture of a typical .NET Core application, including concepts like middleware, DI (Dependency Injection), and the request pipeline.
3. **Performance Optimization Techniques**: Discuss various techniques for optimizing the performance of .NET Core applications, such as asynchronous programming, caching strategies, and tuning garbage collection.
4. **Dependency Injection in .NET Core**: Probe the candidate's understanding of Dependency Injection in .NET Core, including its benefits, usage, and scenarios where it might be inappropriate.
5. **Containerization and Microservices**: Inquire about the candidate's experience with containerization technologies like Docker and how they've been used in conjunction with .NET Core for building microservices architectures.
6. **Security in .NET Core**: Discuss security best practices in .NET Core applications, including authentication (JWT, OAuth), authorization, data protection (encryption, hashing), and securing APIs.
7. **Testing in .NET Core**: Explore the candidate's knowledge of testing methodologies and frameworks in .NET Core, including unit testing with xUnit or NUnit, integration testing, and mocking frameworks like Moq.
8. **Continuous Integration/Continuous Deployment (CI/CD)**: Ask about the candidate's experience setting up CI/CD pipelines for .NET Core projects, including tools like Azure DevOps, Jenkins, or GitHub Actions.
9. **.NET Core vs. .NET Framework**: Discuss the differences between .NET Core and .NET Framework, including their target platforms, performance, features, and migration strategies.
10. **Entity Framework Core**: Assess the candidate's familiarity with Entity Framework Core, including database migrations, LINQ queries, performance considerations, and best practices for working with EF Core in large-scale applications.
11. **Monitoring and Logging**: Inquire about the candidate's experience implementing monitoring and logging solutions in .NET Core applications, such as using frameworks like Serilog or integrating with monitoring tools like Application Insights or Prometheus.

12. **ASP.NET Core Middleware**: Can you explain the concept of middleware in ASP.NET Core? Provide examples of built-in middleware and discuss how custom middleware can be implemented.

13. **Background Tasks in .NET Core**: Discuss different approaches for implementing background tasks or scheduled jobs in .NET Core applications, such as hosted services, Hangfire, or Azure Functions.
14. **Distributed Caching**: How would you implement distributed caching in a .NET Core application for improved performance and scalability? Discuss the use of libraries like Redis or Memcached.
15. **Authentication and Authorization**: Explain the difference between authentication and authorization in the context of ASP.NET Core. Discuss various authentication schemes and authorization policies available in ASP.NET Core.
16. **Cross-platform Development**: How does .NET Core facilitate cross-platform development? Discuss your experience with deploying and maintaining .NET Core applications on different operating systems such as Windows, Linux, and macOS.
17. **High Availability and Scalability**: How would you design a .NET Core application to achieve high availability and scalability? Discuss strategies like load balancing, horizontal scaling, and fault tolerance.
18. **.NET Core CLI Tools**: What are some commonly used .NET Core CLI tools, and how do they aid in development tasks such as project scaffolding, package management, and code analysis?
19. **GraphQL with .NET Core**: Have you worked with GraphQL in .NET Core applications? Discuss its advantages over RESTful APIs, and explain how you would integrate GraphQL into a .NET Core project.
20. **Serverless Computing with .NET Core**: What are your thoughts on using .NET Core for serverless computing? Discuss serverless platforms like Azure Functions or AWS Lambda and their integration with .NET Core.
21. **Error Handling and Logging Strategies**: Describe your approach to error handling and logging in .NET Core applications. Discuss logging frameworks, exception handling techniques, and strategies for handling errors in distributed systems.

21. **Advanced Dependency Injection**: Explain the concept of scoped services in ASP.NET Core DI container. Discuss scenarios where transient or singleton services might not be suitable and why scoped services are preferred.
22. **WebSockets in ASP.NET Core**: Discuss the use of WebSockets in ASP.NET Core applications. Explain how WebSockets differ from traditional HTTP requests and how they can be implemented to enable real-time communication.
23. **GraphQL vs. REST**: Compare and contrast GraphQL with traditional RESTful APIs. Discuss the advantages and disadvantages of each approach and scenarios where one might be more suitable than the other.

24. **Async/Await Best Practices**: What are some best practices for using async/await in .NET Core applications? Discuss common pitfalls to avoid, such as deadlocks, and strategies for handling exceptions in asynchronous code.
25. **Container Orchestration**: Have you worked with container orchestration platforms like Kubernetes or Docker Swarm? Discuss their role in deploying and managing containerized .NET Core applications and the benefits they offer in terms of scalability and resilience.
26. **Health Checks in ASP.NET Core**: Explain the concept of health checks in ASP.NET Core applications. Discuss their importance for monitoring application health and how they can be implemented to provide insights into the application's status.
27. **API Versioning**: How would you handle API versioning in a .NET Core Web API? Discuss different approaches, such as URI versioning, query string versioning, or header versioning, and their pros and cons.
28. **Server-Side Blazor**: What is Server-Side Blazor, and how does it differ from Client-Side Blazor? Discuss the architecture of Server-Side Blazor applications and scenarios where it might be preferred over Client-Side Blazor.
29. **Data Access Performance Optimization**: Discuss strategies for optimizing data access performance in .NET Core applications. Topics may include query optimization, indexing, caching, and minimizing database roundtrips.
30. **Machine Learning with .NET Core**: Have you integrated machine learning models into .NET Core applications? Discuss the use of libraries like ML.NET for tasks such as classification, regression, or anomaly detection within .NET Core applications.

31. **Message Brokers and Event-Driven Architectures**: Discuss the role of message brokers such as RabbitMQ or Apache Kafka in building event-driven architectures with .NET Core. Explain how messages are produced, consumed, and processed within such systems.
32. **Domain-Driven Design (DDD) Principles**: Can you explain how you've applied Domain-Driven Design principles in .NET Core projects? Discuss concepts like bounded contexts, aggregates, domain events, and repositories within the context of DDD.
33. **Immutable Data Structures in C#**: Discuss the benefits of using immutable data structures in C#/.NET Core applications. Explain scenarios where immutability is advantageous and how libraries like Immutable Collections or functional programming techniques can be leveraged.
34. **.NET Core Security Headers**: What are security headers, and how can they be configured in ASP.NET Core applications to enhance security? Discuss common security headers like Content-Security-Policy (CSP), X-Content-Type-Options, and X-Frame-Options.

35. **IdentityServer4**: Have you implemented authentication and authorization using IdentityServer4 in .NET Core applications? Discuss the role of IdentityServer4 as an OpenID Connect and OAuth 2.0 framework and its integration with ASP.NET Core for building secure authentication systems.
36. **Performance Monitoring and Profiling**: How do you monitor and profile the performance of .NET Core applications? Discuss tools and techniques for identifying performance bottlenecks, memory leaks, and CPU usage issues in production environments.
37. **Asynchronous Messaging Patterns**: Discuss common asynchronous messaging patterns like publish-subscribe (Pub/Sub), request-reply, and message queues. Explain how these patterns can be implemented using messaging frameworks like MassTransit or Azure Service Bus in .NET Core applications.
38. **Functional Programming in C#**: Can you explain how functional programming concepts like immutability, higher-order functions, and pure functions are applied in C#/.NET Core development? Discuss the benefits of functional programming paradigms and when to use them.
39. **OpenAPI/Swagger Integration**: How would you integrate OpenAPI/Swagger documentation into a .NET Core Web API project? Discuss the benefits of using OpenAPI/Swagger for API documentation, client generation, and testing.
40. **Concurrency and Parallelism in .NET Core**: Discuss techniques for implementing concurrency and parallelism in .NET Core applications to maximize resource utilization and improve performance. Topics may include asynchronous programming, parallel LINQ (PLINQ), and concurrent collections.
41. **Cross-Cutting Concerns**: How do you address cross-cutting concerns such as logging, caching, and exception handling in a .NET Core application? Discuss the use of aspect-oriented programming (AOP) techniques like interceptors or decorators to modularize these concerns.
42. **GraphQL Subscriptions**: Explain how GraphQL subscriptions work and how they enable real-time data updates in .NET Core applications. Discuss libraries like Hot Chocolate that provide support for GraphQL subscriptions in ASP.NET Core.
43. **Health Checks in Docker Containers**: How would you implement health checks for Docker containers hosting .NET Core applications? Discuss strategies for defining custom health checks and configuring Docker health checks to monitor application health.
44. **Azure Functions with .NET Core**: Have you developed serverless functions using Azure Functions and .NET Core? Discuss how Azure Functions can be used for event-driven architecture, integrating with other Azure services, and handling scale dynamically.

45. **OAuth 2.0 and OpenID Connect**: Explain the differences between OAuth 2.0 and OpenID Connect, and how they are used for authentication and authorization in .NET Core applications. Discuss the role of IdentityServer4 in implementing OAuth 2.0 and OpenID Connect protocols.
46. **GraphQL Federation**: What is GraphQL federation, and how does it enable building distributed GraphQL schemas? Discuss the concept of a gateway schema, federated schemas, and how they can be implemented in .NET Core using tools like Apollo Federation.
47. **.NET Core Performance Counters**: How would you use performance counters to monitor the performance of a .NET Core application? Discuss the types of performance counters available, how to create custom performance counters, and tools for analyzing performance data.
48. **Chaos Engineering**: Have you practiced chaos engineering in .NET Core applications? Discuss the principles of chaos engineering, techniques for injecting faults into distributed systems, and tools like Chaos Toolkit or Gremlin for conducting chaos experiments.
49. **Event Sourcing and CQRS**: Explain the concepts of Event Sourcing and Command Query Responsibility Segregation (CQRS) in .NET Core applications. Discuss how these patterns enable building scalable, event-driven systems and their implementation using frameworks like EventFlow or Akka.NET.
50. **API Gateway with Ocelot**: Have you used Ocelot as an API gateway in .NET Core microservices architectures? Discuss how Ocelot can be used to aggregate, route, and secure API requests, and its integration with service discovery mechanisms like Consul or Eureka.
51. **gRPC in .NET Core**: What is gRPC, and how does it differ from traditional RESTful APIs? Discuss the benefits of using gRPC for inter-service communication in .NET Core microservices architectures, including performance improvements and support for bidirectional streaming.
52. **Blazor Server vs. Blazor WebAssembly**: Compare and contrast Blazor Server and Blazor WebAssembly. Discuss their architectural differences, performance considerations, and scenarios where each approach is preferred.
53. **.NET MAUI (Multi-platform App UI)**: What is .NET MAUI, and how does it simplify cross-platform app development with .NET Core? Discuss its architecture, support for native UI controls, and compatibility with desktop, mobile, and web platforms.
54. **.NET Core Performance Profiling with dotTrace or PerfView**: How would you use tools like dotTrace or PerfView to profile the performance of a .NET Core application? Discuss their features for analyzing CPU usage, memory allocations, and thread contention issues.
55. **Azure DevOps YAML Pipelines**: Have you used Azure DevOps YAML pipelines for CI/CD in .NET Core projects? Discuss how YAML pipelines are defined, version-controlled, and executed as code, and their advantages over classic UI-based pipelines.

56. **Durable Functions in Azure**: What are Durable Functions, and how do they enable stateful, serverless workflows in Azure? Discuss their programming model, support for orchestrator and activity functions, and integration with Azure storage services.
57. **ML.NET Model Serving with ONNX**: How do you deploy ML.NET models using the Open Neural Network Exchange (ONNX) format for interoperability with other ML frameworks? Discuss the process of converting ML.NET models to ONNX format and serving them in .NET Core applications.
58. **ASP.NET Core Health Checks with Kubernetes Probes**: How would you implement Kubernetes readiness and liveness probes using ASP.NET Core health checks? Discuss their role in container orchestration and ensuring the availability of .NET Core applications in Kubernetes clusters.
59. **GraphQL Federation with Hot Chocolate**: Discuss how Hot Chocolate supports GraphQL federation for composing distributed schemas in .NET Core applications. Explain the concepts of schema stitching, remote schemas, and entity resolution in federated GraphQL architectures.
60. **Azure Functions Durable Entities**: What are Durable Entities in Azure Functions, and how do they enable stateful serverless computations? Discuss their usage for managing stateful entities, implementing actor-based patterns, and coordinating distributed workflows in .NET Core applications.
61. **Blazor Hybrid Apps**: Discuss the concept of Blazor hybrid apps, which combine web and native desktop/mobile experiences. Explain how Blazor can be used to build cross-platform desktop/mobile applications with .NET Core and Xamarin.
62. **gRPC Web for Browser Clients**: How would you use gRPC Web to enable communication between browser-based clients and .NET Core services? Discuss the challenges and considerations involved in using gRPC Web compared to traditional gRPC.
63. **Distributed Tracing with OpenTelemetry**: What is OpenTelemetry, and how does it facilitate distributed tracing in .NET Core applications? Discuss its role in monitoring and debugging microservices architectures and its integration with observability platforms like Jaeger or Zipkin.
64. **Azure SignalR Service**: How would you leverage Azure SignalR Service in .NET Core applications for real-time communication? Discuss its benefits for scaling SignalR applications, handling persistent connections, and integrating with Azure services like Azure Functions or Azure App Service.
65. **.NET Core Dependency Injection Improvements**: Discuss the improvements introduced in .NET Core regarding dependency injection, such as support for named dependencies, service provider validation, and better integration with third-party containers like Autofac or Simple Injector.
66. **Blazor PWA (Progressive Web Apps)**: Explain how Blazor can be used to build Progressive Web Apps (PWAs) that provide offline capabilities, push

notifications, and home screen installation. Discuss the features of Blazor PWA templates and their deployment considerations.

67. **Event-driven Microservices with EventGrid**: How would you implement event-driven communication between microservices using Azure Event Grid in .NET Core applications? Discuss the advantages of using Event Grid for decoupling services and handling event routing and filtering.
68. **Machine Learning Operations (MLOps)**: Discuss the principles of MLOps and how they apply to deploying and managing machine learning models in .NET Core applications. Explain techniques for versioning, testing, and monitoring ML models in production environments.
69. **ASP.NET Core WebHooks**: What are ASP.NET Core WebHooks, and how can they be used to implement HTTP callbacks in .NET Core applications? Discuss their role in integrating with third-party services and handling event notifications.
70. **Azure Functions Custom Handlers**: How would you use custom handlers in Azure Functions to extend the runtime with support for additional programming languages or execution environments? Discuss scenarios where custom handlers might be necessary and their integration with .NET Core projects.