SOLID principles

SOLID is an acronym of next items:

S - Single Responsibility Principle

A class should have one, and only one, a reason to change.
One class should only serve one purpose, this does not imply that each class should have only one method but they should all relate directly to the responsibility of the class. All the methods and properties should all work towards the same goal. When a class serves multiple purposes or responsibility then it should be made into a new class.

O - Open-closed Principle

Software entities (classes, modules, functions, etc.) be extendable without actually changing the contents of the class you're extending. If we could follow this principle strongly enough, it is possible to then modify the behavior of our code without ever touching a piece of the original code.

L - Liskov Substitution Principle
Functions that use pointers of references to base classes must be able to use objects of derived classes without knowing it.Subclass/derived class should be substitutable for their base/parent class.

I - Interface Segregation Principle
A Client should not be forced to implement an interface that it doesn't use. This rule means that we should break our interfaces in many smaller ones, so they better satisfy the exact needs of our clients.

D - Dependency Inversion Principle
High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions. Strictly speaking, depend on Abstractions, not on concretions

Design patterns
EF Core
Validation
JWT token
Async/await
Exceptions
LINQ

Reflection
Identity server

Questions above (from **Design patterns** to **Identity server**) cannot be spelled out shortly since skill level is much higher, it is impossible for developers to answer to these points in two words. You find all answers easily if you plan to interview middle/senior developers.

Explain the ASP.NET page life cycle in brief. ASP.NET goes through a series of stages in the life cycle of each page.

- Page request. The user requests a page. ASP.NET decides whether to compile it or serve it from a cache.
- Start. The page's beginning conditions are set.
- Initialization. On-page controls become available, and any themes are applied.
- Load. ASP.NET uses the view state and control state properties to set the control properties.
- Postback event handling. When applicable, user input is handled.
- Rendering. ASP.NET saves the view state for the page and writes the output of rendering to the output stream.
- Unload. The rendered page gets sent to the client. ASP.NET unloads page properties and performs the cleanup.

What is view state in ASP.NET?
View state is data used to preserve page values and control values of Web Forms during postbacks.

What's the difference between Server.Transfer and Response.Redirect.
Server.Transfer sends information from one web request to another, all on the server side. A response is not sent to the browser to cause the change. On the other hand, Response.Redirect sends an HTTP 302 message to the browser and causes a redirect in the browser.

What are the ASP.NET session state modes?

There are several different session state modes in ASP.NET. They provide different ways to store the session state.

1. InProc mode is the default mode. It stores the session state in the web server memory.
2. StateServer mode stores session state in a process known as the ASP.NET state service. If the app is restarted, the session state is preserved.

3. SQLServer mode puts the session state in a SQL database. The session state is preserved if the web app restarts.
4. Custom mode lets the developer specify a specialized storage provider.
5. Off mode disables the session state.

List some different ASP.NET validators.

- Range Validator
- Required Field Validator
- Compare Validator
- Regular Expression Validator
- Custom Validator
- Summary Validator

Explain the difference between Task and Thread in .NET

- Thread represents an actual OS-level thread, with its own stack and kernel resources. Thread allows the highest degree of control; you can Abort() or Suspend() or Resume() a thread, you can observe its state, and you can set thread-level properties like the stack size, apartment state, or culture. ThreadPool is a wrapper around a pool of threads maintained by the CLR.
- The Task class from the Task Parallel Library offers the best of both worlds. Like the ThreadPool, a task does not create its own OS thread. Instead, tasks are executed by a TaskScheduler; the default scheduler simply runs on the ThreadPool. Unlike the ThreadPool, Task also allows you to find out when it finishes, and (via the generic Task) to return a result.