Много способов.

https://question-it.com/questions/1659743/kak-zaregistrirovat-neskolko-realizatsij-odnogo-i-togo-zhe-interfejsa-v-aspnet-core

https://www.stackfinder.ru/questions/39174989/how-to-register-multiple-implementations-of-the-same-interface-in-asp-net-core

Через фабрику

```
services.AddFactory<IProcessor, string>()
                        .Add<ProcessorA>("A")
                        .Add<ProcessorB>("B");
public MyClass(IFactory<IProcessor, string> processorFactory)
{
        var x = "A"; //some runtime variable to select which object to
create
        var processor = processorFactory.Create(x);
}
```

```
public class FactoryBuilder<I, P> where I : class
{
        private readonly IServiceCollection _services;
        private readonly FactoryTypes<I, P> _factoryTypes;

        public FactoryBuilder(IServiceCollection services)
        {
                _services = services; _factoryTypes = new FactoryTypes<I,
P>();
        }
        public FactoryBuilder<I, P> Add<T>(P p) where T : class, I
        {
                _factoryTypes.ServiceList.Add(p, typeof(T));
                _services.AddSingleton(_factoryTypes);
                _services.AddTransient<T>();

                return this;
        }
}

public class FactoryTypes<I, P> where I : class
{
        public Dictionary<P, Type> ServiceList { get; set; }
                                                = new Dictionary<P, Type>
();
}
```

```
public interface IFactory<I, P>
{
        I Create(P p);
}

public class Factory<I, P> : IFactory<I, P> where I : class
{
        private readonly IServiceProvider _serviceProvider;
        private readonly FactoryTypes<I, P> _factoryTypes;

        public Factory(IServiceProvider serviceProvider, FactoryTypes<I,
P> factoryTypes)
        {
                 _serviceProvider = serviceProvider; _factoryTypes =
factoryTypes;
        }
        public I Create(P p)
        {
                return
(I)_serviceProvider.GetService(_factoryTypes.ServiceList[p]);
        }
}

//расширение
namespace Microsoft.Extensions.DependencyInjection
{
public static class DependencyExtensions
        {
                public static IServiceCollection AddFactory<I, P>(this
IServiceCollection services, Action<FactoryBuilder<I, P>> builder) where I
: class
                {
                        services.AddTransient<IFactory<I, P>, Factory<I,
P>>();
                        var factoryBuilder = new FactoryBuilder<I, P>
(services);
                        builder(factoryBuilder);
                        return services;
                }
        }
}
```

Через IEnumerable

```
// In Startup.cs
public void ConfigureServices(IServiceCollection services)
{
        services.AddScoped(IService, ServiceA);
        services.AddScoped(IService, ServiceB);
        services.AddScoped(IService, ServiceC);
}
// Any class that uses the service(s)
public class Consumer
{
         private readonly IEnumerable<IService> _myServices;
         public Consumer(IEnumerable<IService> myServices)
         {
                 _myServices = myServices;
         }
         public UseServiceA()
         {
                 var serviceA = _myServices.FirstOrDefault(t =>
t.GetType() == typeof(ServiceA));
                 serviceA.DoTheThing();
         }
         public UseServiceB()
         {
                 var serviceB = _myServices.FirstOrDefault(t =>
t.GetType() == typeof(ServiceB));
                 serviceB.DoTheThing();
         }
         public UseServiceC()
         {
                 var serviceC = _myServices.FirstOrDefault(t =>
t.GetType() == typeof(ServiceC));
                 serviceC.DoTheThing();
         }
 }
```

Через типизированный интерфейс

**

через типизированный интерфейс
public interface IMyInterface where T : class, IMyInterface

**

#DI, #IOC, #DIP, #asp, #зависимость, #инжекция