There's such a thing as *captured dependencies*. This means that a service lives longer than expected.

So why is that bad?

Well, you want services to live according to their lifetime, otherwise, we take up unnecessary space in memory.

How does it happen?

When you start depending on a Service with a shorter lifetime than yourself you are effectively *capturing* it, forcing it to stay around according to your lifetime. Example:

You register a `ProductsService` with a scoped lifetime and an `ILogService` with a transient lifetime. Then you inject the `ILogService` into the `ProductsService` constructor and thereby *capturing* it.

```
class ProductsService
{
  ProductsService(ILogService logService)
  {

  }
}
```

Don't do that!

If you are going to depend on something ensure that what you inject has an equal or longer life time than yourself. So either change what you depend on or change the lifetime of your dependency.

We complete by declaring the IUnitOfWork type as a "*Scoped Service*" inside the Startup.cs - since DbContext can only be passed onto a transient or scoped service; because DbContext service itself is a scoped ones.