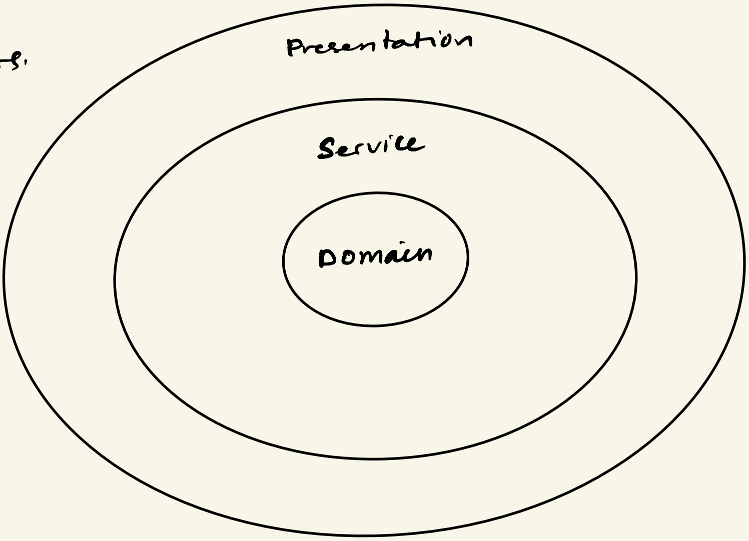
Onion architecture is a form of layered architecture & we can visualize these layers as Concentric layers.

- Usually have 4 layers.
- Domain layer
- Service layer
- Infrastructure layer
- Presentation Layer



Advantages of onion architecture.

- All of the layers interact with each other strictly through the interfaces defined in the layers below. the flow of dependencies is towards the core of the onion.
- Using dependency inversion throughout the project, depending on abstractions (interfaces) & not the implementations, allows us to switch out the implementation at runtime transparently.
- We are depending on abstractions at compile-time, which gives us strict contracts to work with, & we are being provided with implementation at runtime.

- Testability is very high with the onion architecture because everything depends on abstractions.

Flow of Dependencies

- the main idea behind the onion architecture is the flow of dependencies, or rather how the layers interact with each other. the deeper the layer resides inside the onion, fewer dependencies it has.

- the Domain layer does not have any direct dependencies on the outside layers. It is isolated, in way, from the outside world. the outer layers are all allowed to reference the layers that are directly below them in the hierarchy.

- we can conclude that all dependencies in onion architecture flow inwards

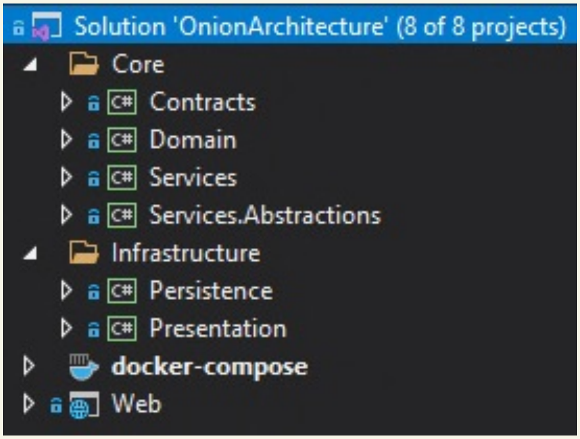
- the flow of dependencies dictates what a certain layer in the onion architecture can do. Because it depends on the layers below it in the hiearchy.

- We can use lower layers of the onion architecture to define contracts or interfaces.

the outer layers of the architecture implement these Interfaces. that means that in Domain layer, we are not concerning ourselves with infrastructure details such as the database or external services.

Using this approach, we can encapsulate all of the rich business logic in the Domain & service layers without ever having to know any implementation details.

In the service layer, we are going to depend on the interfaces that are defined by the layer below, which is the Domain layer



```
Solution 'OnionArchitecture' (8 of 8 projects)
  Core
    Contracts
    Domain
    Services
    Services.Abstractions
  Infrastructure
    Persistence
    Presentation
  docker-compose
  Web
```

As you can see, It consists of the Web Project, & six class libarics. the Domain Project will hold the Domain layer implementation. the Services & Services. Abstractions are going to be our service layer implementation. the Persistence Project will be our infrastructure layer, & Presentation Project will be the Presentation layer Implementation.