1. **Name examples of the layered architecture. Do they differ or just extend each other?**

* DDD (Domain Driven Design).
* Clean Architecture.

Both architectures are very similar but differ in some parts, for example in DDD there is a Domain Service that is a specific type of domain layer class what is used when it is needed to put some domain logic that relies on two or more entities. In clean Architecture this logic is placed into the application service.

1. **Is the below layered architecture correct and why? Is it possible from C to use B? from A to use C?**

Chart, box and whisker chart

Description automatically generated

It is possible since layers can be opened, but int the example it is experiencing the sinkhole anti-pattern that makes more difficult to control change due to the lack of layer isolation.

1. **Is DDD a type of layered architecture? What is Anemic model? Is it really an antipattern?**

Dirven Domain Design is a type of layered architecture with a rich domain model in the center.

Anemic model is the opposite of the DDD architecture. And it is considered an antipattern where the domain objects (entities) don’t have any business logic. This business logic is part of the program itself, making refactoring and maintenance more difficult and time-consuming.

1. **What are architectural anti-patterns? Discuss at least three, think of any on your current or previous projects**.

*The blob*. I have encountered this on several occasions. In the current project I’m working there are some classes with lot of responsibilities. I think this is caused because the code was touched by multiple developers for a long time. Anyone wanted to split the code in different classes and now we end up with this kind of antipatterns.

*Continuous of obsolescence*. It is a kind of common antipatterns that I have experienced. In fact, it is very complicated to find a project with the last version of .NET. I should say that I’m not experienced in .Net 6 because I have made some projects in new version just for curiosity. In my current project I’m working with the asp framework. And in past projects it was the same with old technologies like working with COM services and so on.

*Spaghetti Code*. This is very common in asp pages when you look at the JavaScript block, you find there a lot of the behavior of the page with lots of functions and poor software structure. I must confess that I have done that too even when I try to avoid it.

1. **What do Testability, Extensibility and Scalability NFRs mean. How would you ensure you reached them? Does Clean Architecture cover these NFRs?**

The are quality attributes that describe how the system should be.

*Testability*. Because components belong to specific layer in the architecture, other layers can be mocked or stubbed, making this relatively easy to test.

*Extensibility.* This architecture allows to extend nonfunctional requirements since the application layer has this purpose.

*Scalability*. Because of the trend toward tightly coupled and monolithic implementations of this pattern, applications build using this architecture pattern are generally difficult to scale. But in mixing Clean Architecture with microservices could give us the scalability needed but losing testability of the whole application.