Software Architecture: Layered Architecture

Made By: Mamunur Roshid

1. Introduction

In this article, we're going to look at the layered architecture in software development. We'll give an overview of what it is, its components, and outline its characteristics. Next, we'll also describe the benefits and drawbacks, and give examples of situations where they are appropriate for use.
Before we begin, let's first define software architecture.

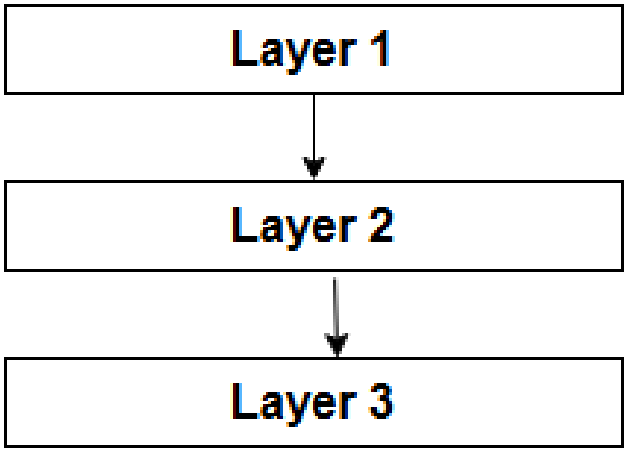
2. Definitions

Software architecture refers to the basic structure of any software system and incorporates any aspect that makes a system function and behaves as it should. Although the term architecture usually refers to physical designs, in software systems, it encompasses the design of components, relationships between components, user interactions, as as well as the user's needs of a system.

There are several different software architectures that exist such as microkernels, microservices, and client-servers just to name a few. Each of these is structured differently and is used in a different context. However, we'll only look at layered architectures in this article.

3. What Is Layered Architecture?

Layered architectures are said to be the most common and widely used architectural framework in software development. It is also known as an n-tier architecture and **describes an architectural pattern composed of several separate horizontal layers that function together as a single unit of software**. A layer is a logical separation of components or code:



In these frameworks, **components that are related or that are similar are usually placed on the same layers**. However, each layer is different and contributes to a different part of the overall system.

3.1. Characteristics

A major characteristic of this framework is that layers are only connected to the layers directly below them. In the illustration given previously, layer 1 only connects to layer 2, layer 2 connects to layer 3, and layer 1 is connected to layer 3 only through layer 2.

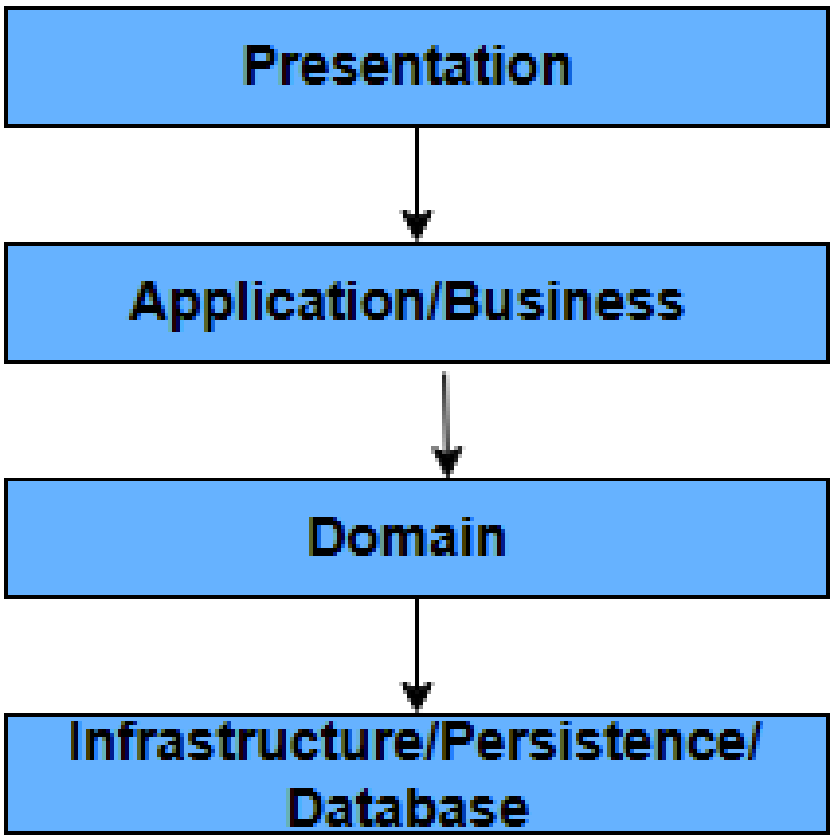
Another characteristic is the concept of **layers of isolation**. **This means that layers can be modified and the change won't affect other layers**. In short, changes are isolated to the specific layer that is altered.

Separation of concerns is another notable feature that speaks to how the modules on a single layer together perform a single function.

4. Components of a Layered Architecture

Now, **the number of layers in a layered architecture is not set to a specific number** and is usually dependent on the developer or software architect. It is important to note that this framework will usually always have a user interaction layer, a layer for processing, and a layer that deals with data processing. These are described further as:

- **Presentation Layer** – responsible for user interactions with the software system
- **Application/Business Layer** – handles aspects related to accomplishing functional requirements
- **Domain Layer** – responsible for algorithms, and programming components
- **Infrastructure/Persistence/Database Layer** – responsible for handling data, databases



Additionally, in some applications, some layers are combined. For example, it is common to find the business layer and persistence layer combined into a single layer. This just means that the functions and responsibilities of these two layers have been grouped to occur at a single layer.

5. Advantages and Disadvantages

The following are the benefits and drawbacks that exist with this software pattern:

Advantages

- The framework is simple and easy to learn and implement.
- There is reduced dependency because the function of each layer is separate from the other layers.
- Testing is easier because of the separated components, each component can be tested individually.
- Cost overheads are fairly low.

Disadvantages

- Scalability is difficult because the structure of the framework does not allow for growth.
- They can be difficult to maintain. A change in a single layer can affect the entire system because it operates as a single unit.
- There is interdependence between layers since a layer depends on the layer above it to receive data.
- Parallel processing is not possible.

6. When to Use Layered Architectures

When developing simple, small applications, it is advisable to implement a layered architecture because it’s the most simple framework. However, some

developers are of the opinion that because they can be difficult to maintain, it is better to apply them to larger projects.

In spite of this, the framework can be used for applications that need to be built quickly because it's easy to learn and implement. It is also good in cases where the developers do not have a lot of knowledge of software architectures or when they are undecided on which one to use.

Some real-life applications of this architecture are in web applications and the OSI model. The J2EE programming model also implements the layered architecture.

7. Conclusions

In this article, we've defined layered architectures. We've also discussed their characteristics and the basic components that exist. We've looked at the benefits and drawbacks and have also detailed scenarios where it is appropriate to use them.