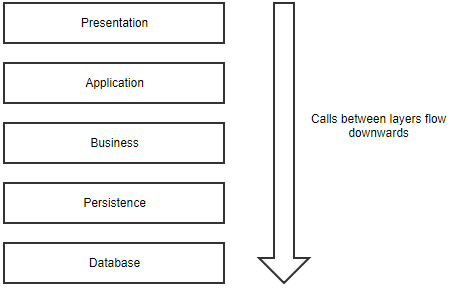
**Software Architecture**

Architecture has some well-known patterns followed in the software world. Those are

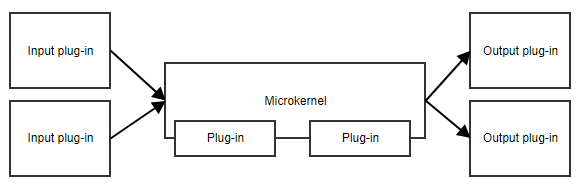
1. Layered Pattern

Probably one of the most well-known software architecture patterns. Idea is to split up your code into layers where each layer has a certain responsibility and provide service to the higher layer.



1. Microkernel or Plug-in pattern

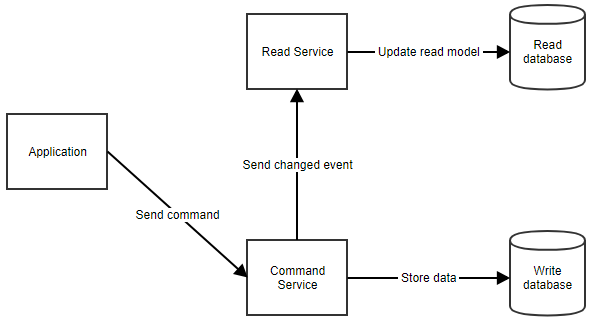
Is useful when your application has a core set of responsibilities and a collection of interchangeable parts on the side. The microkernel will provide the entry point and the general flow of the application, without really knowing what the different plug-ins are doing.



A suitable example is Task scheduler or a workflow.

1. CQRS (Command and Query responsibility Segregation)

The central concept of this pattern is that an application has read operations and write operations that must be totally separated. This also means that the model used for write operations (commands) will differ from the read models (queries).



Ideal for applications that expect a high amount of reads

1. Event Sourcing

This is a pattern where you don’t store the current state of your model in the database, but rather the events that happened to the model. So, when the name of a customer changes, you won’t store the value in a “Name” column. You will store a “NameChanged” event with the new value (and possibly the old one too).

A real-life analogy of event sourcing is accounting.

This software architecture pattern can provide an audit log out of the box. Each event represents a manipulation of the data at a certain point in time.

1. Microservices

When you write your application as a set of microservices, you’re actually writing multiple applications that will work together. Each microservice has its own distinct responsibility and teams can develop them independently of other microservices. The only dependency between them is the communication.

