Software architecture evolution

In this project we are going to explain the main phases the software applications have gone through during their existence, starting from the oldest and simplest, the standalone systems to the more complex ones, microservices.

One tier architecture

One Tier application is also known as a Standalone application. It is equivalent to running the application on a personal computer, where all the required components for the application to run are in a single server or application. Some well known apps that follow this approach are MS office or the MP3 player.

Its advantages are the efficiency, that it is easy to implement and the fact of omitting context switching issues. Its main disadvantage is that it does not support remote or distributed access for data resources.

Two-tier architecture

The two-tier architecture is also known as client-server application. It is divided in two parts, the client application and the server. The client system handles Presentation layer, and the Server system handles the Database and Business layer. An example of this type of apps would be a banking app, where the client app communicates through an internet connection to the Bank Server.

Its advantages are that it is easier to maintain and making modifications its not difficult at all and that the communication is quite fast. Its main disadvantage is that a large number of users may decrease the performance of the application.

Three Tier Architecture

Three Tier application is also known as Web Based application. These kinds of applications are divided in three layers, presentation, application and data storage. In this architecture the client system handles the presentation layer, the application server handles the application layer, and the server system handles the database layer.

Its advantages are the higher security because the client does not have direct access to the database, and its high scalability because each layer runs on a different server. Its main disadvantage is that the structure is more complex comparing it with the previous ones.

Web Servers

The web server works as a giant computer where the content of the web is stored. The basic objective of the web server is to store, process and deliver web pages to the users. Web servers can run static content, as html, css, js, images and pdf files. The communication between the user and the web server is done using the http protocol. Some examples of these web servers can be the Apache HTTP Server o the Microsoft Internet Information Services(IIS).

Web Containers

Web containers can do what web server can do addition to that it can serve dynamic content. This means webs containers can run dynamic content as well as static content. The main difference between both is that in a web container the content of the web page can change.

Application Server

This server is specifically designed to run applications. An application server provides the processing power and memory to run applications in real time. Application servers are more powerful and faster than web servers, but they are much more expensive as well.

Service Oriented Architecture (SOA)

These applications use the services that are available in the network, which are provided through a communication call over the internet. This allows us to reuse any services in our application. These services use a common interface and an architectural pattern so they can be incorporated into new applications.

In SOA there are two main roles, the service provider is the maintainer of one or more services for others to use it, and the service consumer which develops the required client components to bind and use the service.

Micro Services

Micro services are self-contained and independent deployment modules. They are created individually and deployed separately from one another. They also are the main architectural design when building a distributed application using containers, this means services can be changed without affecting other parts of the application.

Some of the key points of microservices are:

- The app is broken into modular and loosely coupled components.
- The app can be distributed across clouds and data centers.
- Adding the features only requires updating those individual microservices.
- Network services must be software-defined and run as a fabric for each microservice to connect to.

Some disadvantages are:

- The increased difficulty when managing a large number of microservices.
- The complexity of testing the microservices over a distributed environment.

Some of the most innovative and profitable companies in the world use microservices, like Amazon, Netflix, or Uber.