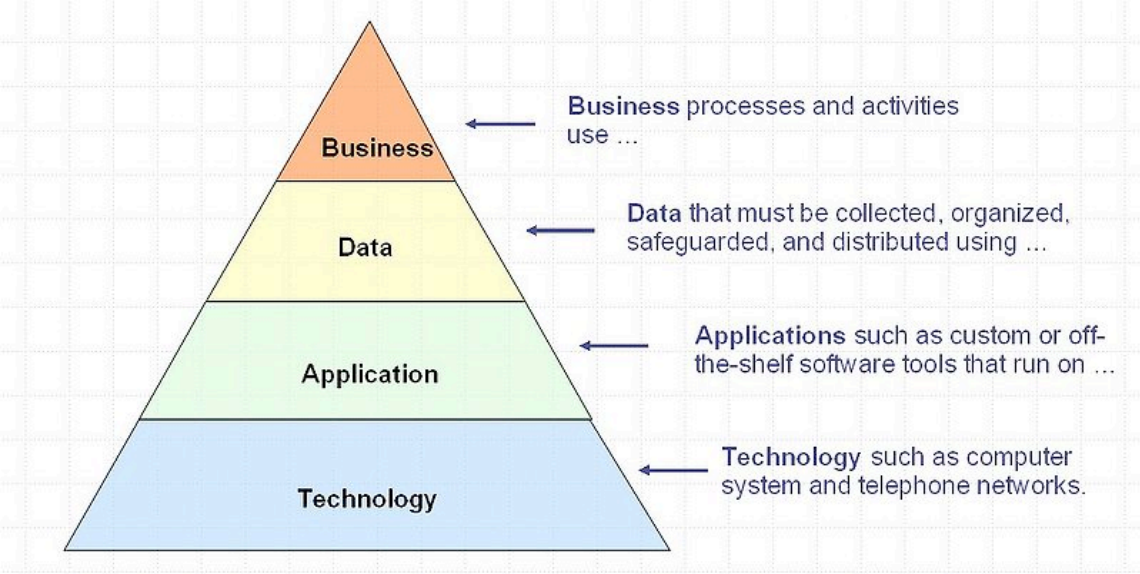
Modelování a návrh architektury software. Klient-Server. Třívrstvá architektura a její zobecnění na vícevrstvou architekturu. Architektonické vzory, Model-View-Controller (MVC). (A7B36WPA)

MODELOVÁNÍ A NÁVRH ARCHITEKTURY

Návrh architektury SW je klíčovým momentem při vývoji aplikace. Softwarový návrhář má možnost využít pro svůj návrh některý z již existujících vzorů. Ty mu pomůžou vytvořit funkční model na již vyskoušených základech. Každý vozr má totiž známe klady a zápory a lze se tak vyvarovat chyb při návrhu aplikace. Je nutné řici, že na většině aplikací se uplatní více softwarových vzorů.

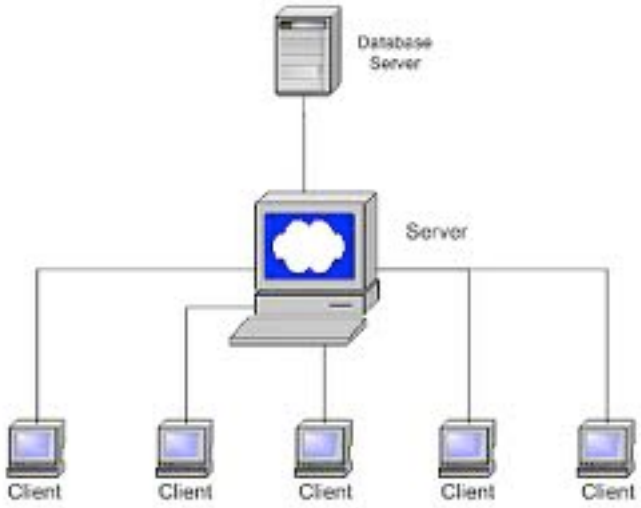


Obrázek vystihuje motivaci nutnosti modelování a návrhu architektury. Žádná z vrstev nemusí vědět nic o tom jak fungují ty další. Musí znát pouze fungování rozhraní mezi nimi. Když se pak používají standardizovaná rozhraní, lze jednotlivé vrstvy nahradit bez toho, aby bylo nutné nahradit celou strukturu aplikace.

ARCHITEKTURA KLIENT - SERVER

Architektura klient-server dovoluje delegovat výpočetní výkon nebo potřebu datového úložiště na server. Na klientské straně tak může běžet jen grafické rozhraní a veškeré další požadavky jsou odesílány na server, který je vyřizuje a případné výsledky posílá zpět klientovi, který je už jen zobrazí.

Při návrhu aplikací této architektury bývá z pravidla nejužším hrdlem spojení mezi serververem a klientem. Proto je třeba toto brát v úvahu a komunikaci mezi klientem a serverem využívat co nejúčiněji.



Výhodou je pak centralizace systému. Aktualizovat a opravovat chyby v aplikační logice serveru je pak velmi jednoduché. Dále se centralizuje datové úložiště a při vhodném návrhu lze pak i ušetřit prostor pro ukládání dat. Data totiž není potřeba ukládat na klientovi.

Obecnou výhodou architektury klient - server jsou nízké nároky, které lze na klienta klást. Ukázovým příkladem jsou mobilní aplikace, kde se tento model uplatňuje nejčastěji. Data uživate jsou uloženy na serveru a v momentu přihlášení se ze serveru data stahují jen v případě potřeby ‘on demand’, na mobilním zařízení jsou cachována a zpět se přenášení jen změny.

ARCHITEKTONICKÉ VZORY

Neplést s návrhovými vzory. Ty jdou více do podrobna. Architektonické vzory jsou obecnější (i když se některé techniky prolínají. viz seznam arch. vzorů). Architektonických vzorů existuje celá řada a obecně definují rozložení složitější aplikace do logických celků, které spolu komunikují. Architektonické vzory mají pomoci jednodušeji navrhnout architekturu vlastní aplikace. Lze se totiž podržet již hotového vzoru. Těch existuje celá řada. Jako první je třeba zmínit vzor ‘monolitické aplikace’, který je prakticky jednovrstvou architekturou. Jde o stupidní architekturu spojující aplikační logiku se zobrazovací a přesto jde o vzor.

EDA je event driven architektura - velmi často používaná v GUI. Funguje tak, že na jednotlivé události se navěsí listenery, které v případě spuštění události (trig) se zavolá navěšená metoda - listener.

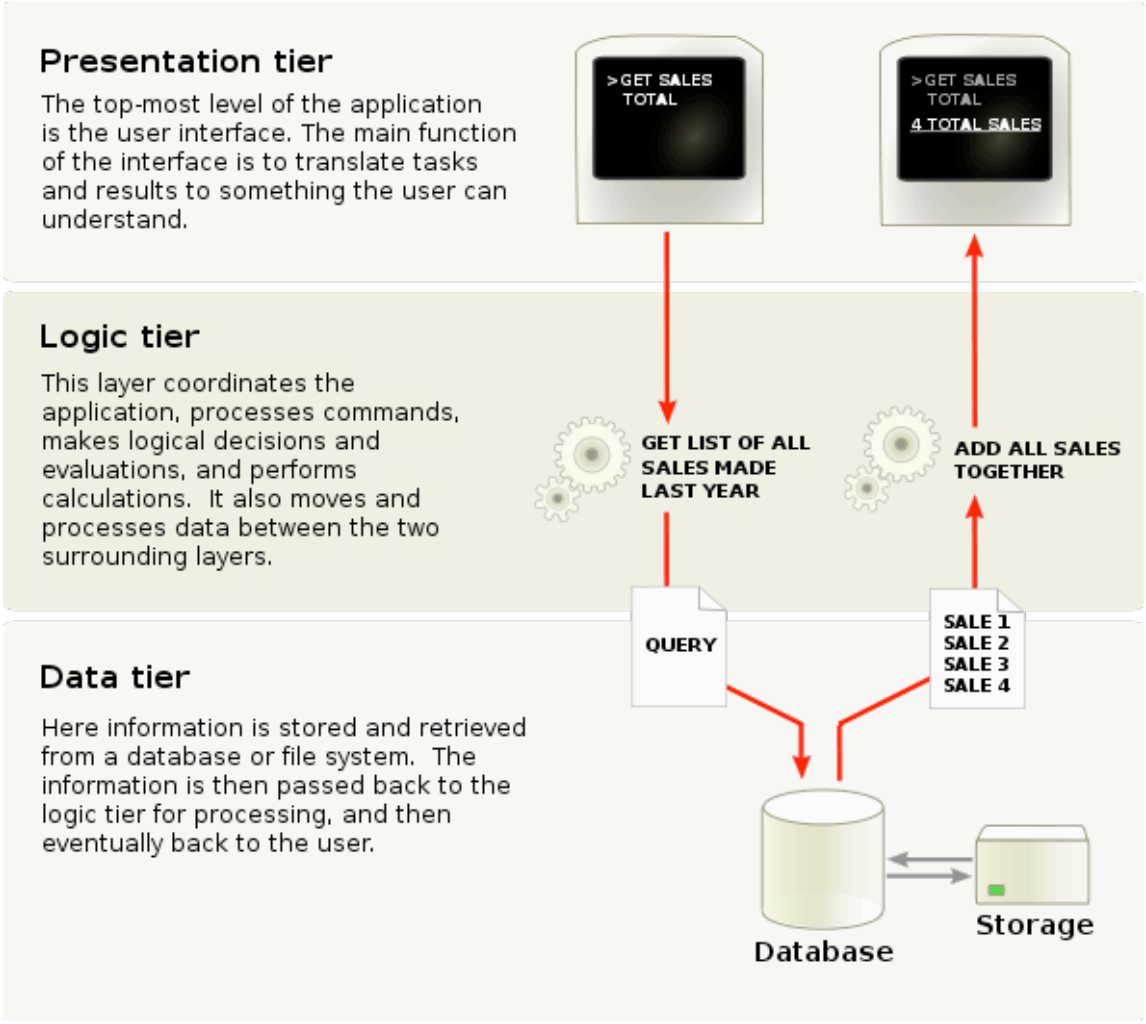
SOA je službově orientovaná architektura. Ta se používá v případech, kdy je třeba propojovat různé platformy. Služby překryjí aplikační logiku a reprezentují konkrétní běžné úlohy, které aplikace vykonává. SOA se často používají ve webových technologiích a existují pro ně standardy jako SOAP.

P2P je architekturou, kdy spolu komunikují navzájem klienti. Každý klient se umí stát serverem. Tato architektura se nejčastěji používá pro výměnu souboru na síti na celosvětové, ale i lokální úrovni.

TŘÍVRSTVÁ ARCHITEKTURA A JEJÍ ZOBECNĚNÍ NA VÍCEVRSTVOU

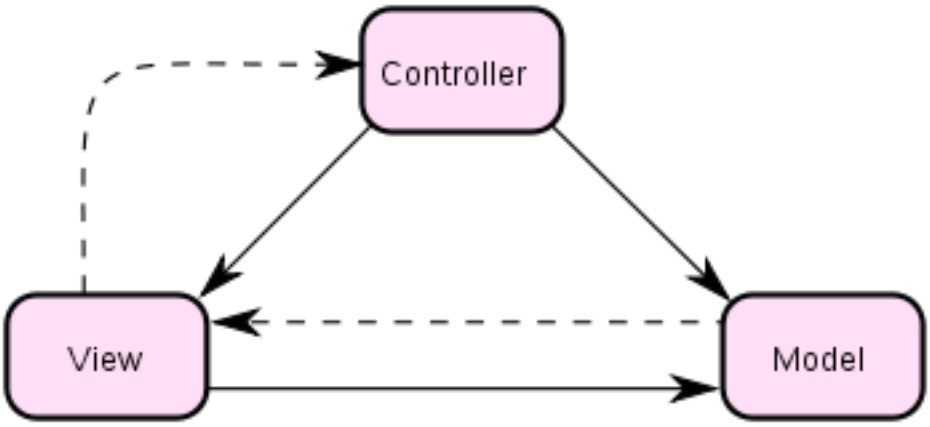
Základním principem třívrstvé architektury je oddělení Datové vrstvy od Prezentační vrstvy vrstvou Aplikační (zvaná také Business vrstvou). Komunikace mezi vrstvami pak probíhá lineárně. Od datové vrstvy, přez aplikační do prezentační a zpět zas přez aplikační do datové.

Třívrstvou architekturu lze zobecnit na vícevrstvou tak, že si jako Datovou vrstvu představíme databázový server, jako Aplikační vrstvu si představíme webový server a jako Prezenční vrstvu si představíme internetový prohlížeč. Je třeba zopakovat že jde o zobecnění. Třívrstvou architekturu můžeme uplatnit třeba jen na jedné vrstvě tohoto zobecnění - například na webovém serveru.



MODEL-VIEW-CONTROLER

Na rozdíl od třívrstvé architektury, která je lineární, MVC architektura je triangulární. Jednotlivé vrstvy spolu komunikují v trojuhelníku. M značí Model - aplikační vrstvu, V značí View - prezenční vrstvu a C značí controller - vrstvu která je prostředníkem mezi View a Modelem. Trojuhelník funguje tak, že View získává data přímo z Modelu a z View se předávají data prostřednictvím Controlleru Modelu.



MVC architektura bývá nejčastěji používána u aplikací s grafickým uživatelským rozhraní. Při dodržení zásad definovaných touto architektirou se kód stává přehlednějším. Odděluje totiž striktně aplikacni logiku od kódu, který se stará o komunikaci s uživatelem - View. O komunikaci mezi aplikacni logikou (Model) a uživatelským rozhraním (View) se stará Controler. Aplikační logika se pak stává znovupoužitelnou, protože se nijak neváže na potřeby grafického rozhraní a grafické rozhraní je možné používat univerzálněji.