**Architectural design patterns**

**- MVC, Multitier, Monolithic, P2P, Client/Server**

**NÁVRHOVÝ VZOR**

* Obecné řešení problému
* Tzv šablona pro návrh programů
* Ukazují vztahy a interakce mezi **třídami** a **objekty**
* **! ALGORYTMY –** nejsou považovány za návrhové vzory
* Nejsou jen z programování, jsou také v každodenním životě
  + Např. architektura (stavby)

**ZÁKLADNÍ TYPY**

* **Creational patterns (vytvářející)**
  + Řešení problémů s vytvářením objektů v systému
  + Popisování postupu výběru třídy nového objektu
  + Dynamická rozhodnutí za běhu programu
* **Structural patterns (strukturální)**
  + Uspořádání jednotlivých tříd nebo komponentů
  + Důvody
    - zpřehlednit systém
    - Strukturalizace kódu
* **Behavioral patterns (chování)**
  + Chování systému
  + Třídy
    - Principy dědičnosti
  + Objekty
    - Spolupráce mezi objekty
    - Spolupráce mezi skupiny objektů
    - Dosažení požadovaného výsledku

**MVC**

* Rozděluje datový **model aplikace / uživatelské rozhraní / řídící logiku**
* může být pojat jako i architektonický vzor nebo agregační návrhový vzor
* Skládá se ze 3 komponent
* **CONTROLLER**
  + Reaguje na události (typicky od uživatele)
  + Zajišťuje změny v modelu
* **MODEL**
  + Reprezentace informací
  + S informacemi aplikace pracuje
  + Je řízeno CONTROLLEREM
* **VIEW**
  + Převádí data z modelu do vhodné podoby
  + Podoba pro interakci/prezentaci uživateli
* Obecný princip:
  + Uživatel provede akci
  + Controller obdrží oznámení o akci
  + Controller v modelu provede potřebné změny
  + View použije obnovený Model pro zobrazení dat uživateli

# Multitier / Vícevrstvá architektura

# Aplikace, který funkce netvoří celistivý program

# Jeho vrstvy vzájemně spolupracují

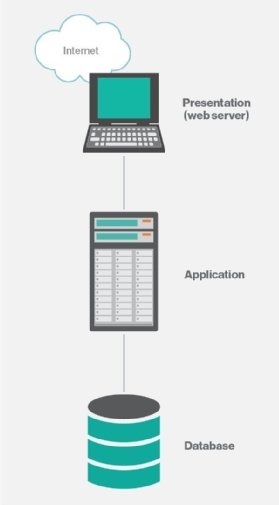
# PŘ: internetový obchod

# klientská část běží v prohlížeči uživatele

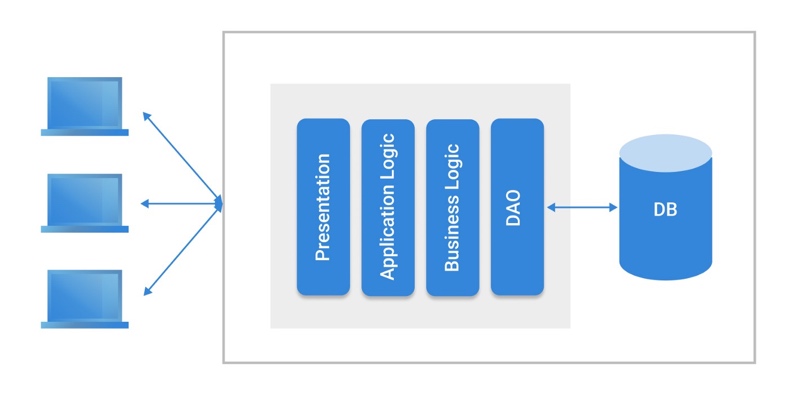
# aplikační logika na webovém či aplikačním serveru

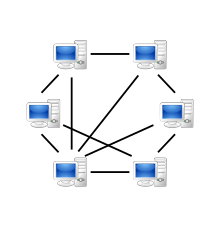
# data sjsou uložena v databázovém serveru.

# THREE TIER

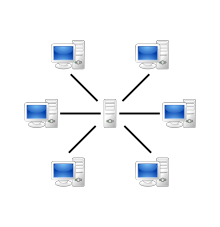
* Client-server architektura
* Každý modul je alokovaný na jiný oddělený platformě
* Jednotlivý moduly mohou být vylepšovány nezávisle na sobě
* **Hlavní výhody**:
  + Dává svobodu pro vývojové týmy -> modifikování specifických částí změny celé aplikace
  + Aplikace je lehce rozšířitelná pomocí např. oddělení frontendu od databáze
  + Zamezení částí aplikace bez změn na běhu programu
  + Možnost adaptace k novým technologiim
  + Různé týmy na různých částí aplikace
* Rozdělení:
  + **Presentation**
    - Vrchní úroveň
    - Zobrazuje informace které prezentují dostupné funkce
    - Tzv. Front-end pro interakci uživatele s aplikací
    - Často stavěn na frameworkách
  + **Application**
    - Prostřední úroveň
      * Logická, buisness úroveň
      * Ovládá hlavní funkce aplikace
  + **Data**
    - Databázová úroveň
    - Oracle, MySQL, MSSQL …

**MONOLITIC**

* Celá aplikace funguje jako jedna vrstva (jedna skupina objektů)
* Způsob se více hodí pro menší aplikace
* Spíše **ANTIPATTERN**
* **Výhody**
  + Lehce vyvijitelná
  + Lehce nasazovatelná (jeden WAR soubor)
  + Lehce rozšiřovatelná
  + Lehce testovatelná
* **Nevýhody**
  + Težká na pochopení novými programátory
    - Zpomalování vývoje
  + Těžké provádění změn (k nepochopení)
  + Přetížení webu -> zpomalení zapínání a nasazování
  + Pravidelné nasazování -> Při malé funkci se nasadí celý program znova

**PEER-TO-PEER**

* Také se označuje Klient-Klient
* Typ počítačové sítě kde jsou klienti přímo připojený na sebe
  + Opak je client-server (komunikace přes centrální server)
* Působí současně jako klienti i servery
* V praxi se však objevují specializované servery, kteří zjednoduší návrhu
  + Slouží pro počáteční navázání
* Dnes je P2P se vztahuje hlavně pro výměnu dat
* Základní výhoda:
  + S rostoucím množstvím uživatelů se přenosná kapacita roste
    - U client-server průměrná rychlost přenosu klesá
* Velká část obsahu šířená přes P2P je **PIRÁTSKÁ**

**KLIENT-SERVER**

* Síťová architektura
* Odděluje klienta a server (server hlavní bod který propojuje)
* Princip:
  + Klient žádá o služby server (prostřednictvím např. prohlížeče)
  + Prohlížeč předá informaci servru
  + Server se dotáže na databázi
  + Výsledek je zpět zaslán stejnou cestou
* Je to jedna z hlavních myšlenek síťové technologie
* Tento typ architektury je občas označován jako „Two-tier“
* **Charakteristiky stran**:
  + **Klient**
    - Je aktivní
    - Posílá žádosti na server
    - Po žádosti čeká a dostává odpovědi
    - Je většinou připojen k malému počtu serveru
  + **Server**
    - Je pasivní
    - Naslouchá na síti a reaguje na žádosti
    - Při přijetí požadavku – obslouží
* Výhody:
  + Většinou rozděluje jednotlivé úkoly
  + Údaje uloženy na serveru – zabezpečení
    - Přímo přistupovat mohou pouze oprávně