

Druhy rozhraní

- Príkazový riadok (CLI - Command Line Interface)
- Ovládanie cez Menu (Menu Driven Interface)
- Grafické (GUI - Graphical User Interface)
- Prirodzený jazyk (Natural Language Interface)
- Iné

Príkazový riadok

➤ Výhody

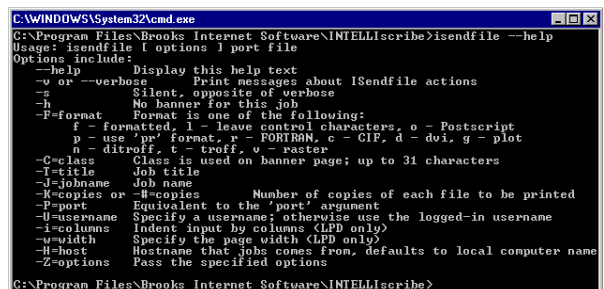
- Efektívne využitie prepínačov
- Vhodné pre expertov
- Nízke požiadavky na zdroje systému

➤ Nevýhody

- Potreba naučiť sa príkazy
- Neviditeľné možnosti
- Nevhodné pre začiatočníkov

➤ Typické aplikácie

- Administrácia systémov (Unix/Linux)
- Inžinierske aplikácie
- Vedecké aplikácie
- Ideálne pre nevidiacich a zrakovo obmedz.



```
C:\WINDOWS\System32\cmd.exe
C:\Program Files\Brooks Internet Software\INTELLiscribe>lsendfile --help
Usage: lsendfile [ options ] port file
Options include:
--help          Display this help text
-v or --verbose Print messages about lsendfile actions
-s             Silent, opposite of verbose
-h            No banner for this job
-P=format      Format is one of the following:
               f - formatted, l - leave control characters, o - Postscript
               p - use 'pr' format, F - FORTRAN, c - CIF, d - dvi, g - plot
               n - ditroff, t - troff, v - raster
-C=class       Class is used on banner page; up to 31 characters
-I=title       Job title
-J=jobname     Job name
-K=copies or -K=copies Number of copies of each file to be printed
-P=port        Equivalent to the 'port' argument
-U=username    Specify a username; otherwise use the logged-in username
-i=columns     Indent input by columns (LPR only)
-w=width       Specify the page width (LPR only)
-h=host        Hostname that jobs comes from, defaults to local computer name
-Z=options     Pass the specified options
C:\Program Files\Brooks Internet Software\INTELLiscribe>
```

Menu

➤ Výhody

- Nie je nutné učiť sa komplexné príkazy/jazyk
- Ľahké použitie i pre nováčikov
- Ideálne pokiaľ je množstvo volieb obmedzené

➤ Nevýhody

- Pre skúsených používateľov zdĺhavé napr. ak sa príkaz nachádza až v 5 podúrovni
- Obmedzené rozmerom obrazovky a max. počtom položiek

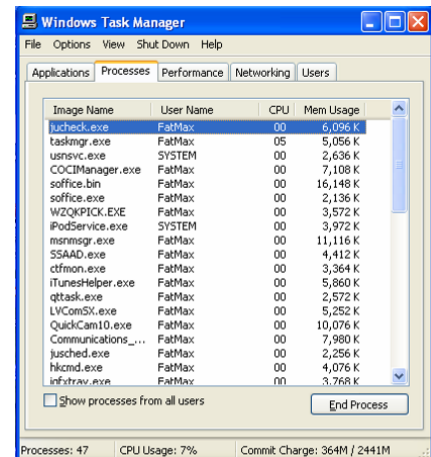
➤ Typické aplikácie

- Bankomat
- Klasické mobilné telefóny
- Domáce spotrebiče
- TV
- Navigačné systémy



Grafické

- Vhodné pre začiatočníkov a pokročilých ale ...
- Vyššie systémové nároky
- Zložitejšia navigácia
- ...
 - Grafické rozhrania často využívajú spoločné prvky
- Rozmiestnenie ovládacích prvkov
- Názvy operácií
- Ikony
- Poradie a obsah menu
- Operácie myšou
- ...
 - Štandardizácia GUI ma za dôsledok
- Zrýchlenie učenia sa používateľov
- Jednoduchosť použitia aplikácií
- Rozšírenie množstva riešiteľných problémov
- Napomáha sebavedomiu nových používateľov
- Väčšie množstvo užitoč. aplikácií pre používateľa

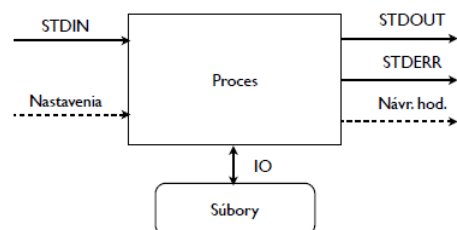


Prirodzený jazyk

- Výhody
- Nie je nutné učiť sa spôsob interakcie
- Častokrát rýchlejšie než klávesnica
- Nie je nutné použiť ruky
- Ľahko použiteľné i pre ľudí s obmedzením
- Nevýhody
- Často krát nedostatočná spoľahlivosť
- Problém s nárečím a s niektorými jazyk. konštrukciami
- Umelé jazyky a obmedzenia sú často presnejšie

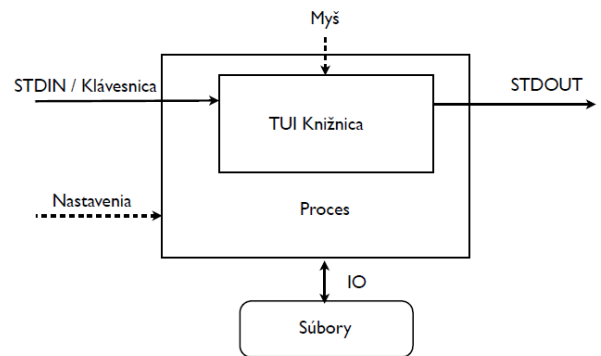
Príkazový riadok

- Jednoduchá architektúra a vysoká prenosnosť
- Typické použitie v kontexte jednoduchých znovupouž. aplikácií operačných systémov
- Efektívne spracovanie textového vstupu a súborov
- Obmedzená priama interakcia
- Aplikácie možno efektívne zreťaziť a kombinovať
- Ľahká interakcia so vzdialeným systémom



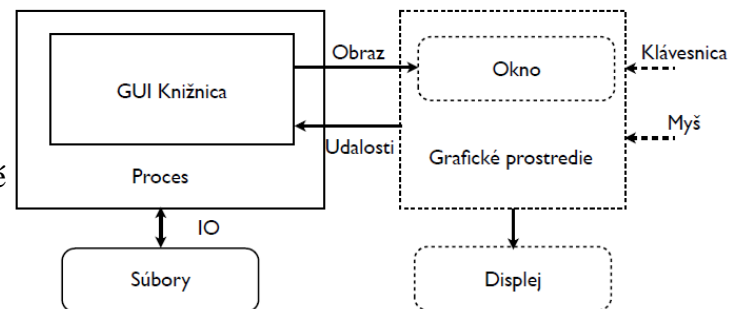
Konzolové aplikácie

- Aplikácie s textovým používateľským rozhraním
- Architektúra porovnateľná s aplikáciami na príkazovom riadku
- Využívajú na interakciu s používateľom textové ovládacie prvky
- Nie je možné ich efektívne zreťaziť a automatizovať
- Možno použiť mnohé knižnice pre tvorbu textových rozhraní (TUI) napr. *curses*, *conio*
- Priama interakcia za pomoci klávesnice a myši
- Jednoduchosť pri práci cez vzdialený prístup

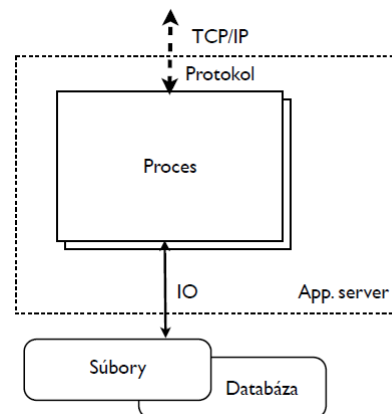


Grafické aplikácie

- Situácia je podobná ako u TUI aplikácií
- Interakcia je sprostredkovaná cez **grafické prostredie** formou zasielania správ
- Aplikácia vytvára **používateľské rozhranie** ktoré chce zobraziť (obraz)
- Používateľské rozhranie generuje GUI knižnica



Server aplikácie



Polia a JavaScript

```
var arr = [ 10, "hello", function(){}, [1, 2, 3]]

for (i=0;i<arr.length;i++) {
  console.log(typeof(arr[i]) + " - " + arr[i])
}
```

```
number - 10
string - hello
function - function (){}
object - 1,2,3
```

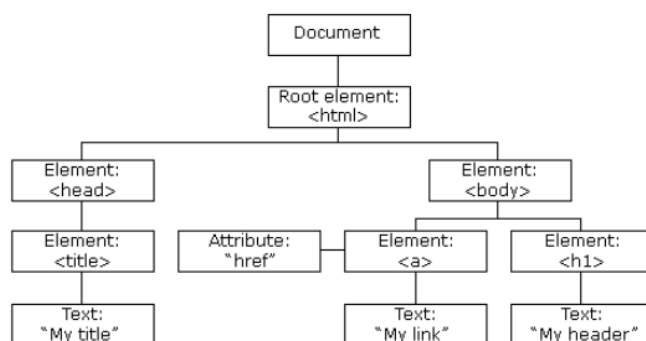
Objekty a JavaScript

- Objekty možno reprezentovať priamo pomocou {}
- Uchovanie pomenovaných atribútov

```
var person = {  
  name: "john",  
  age: 27,  
  scream: function() {console.log("Arrrgh")}  
}  
console.log(person.name)  
console.log(person["age"])  
person.scream()
```

DOM (Document Object Model)

- Reprezentácia načítaného dokumentu pomocou JavaScript objektov
- Do dokumentu možno priamo zapisovať
- Je možné element uložiť do premennej
- Môžeme manipulovať štýl zobrazenia
- Môžeme vykonať kód pri kliknutí myšou
- Elementy možno i vytvárať a mazať

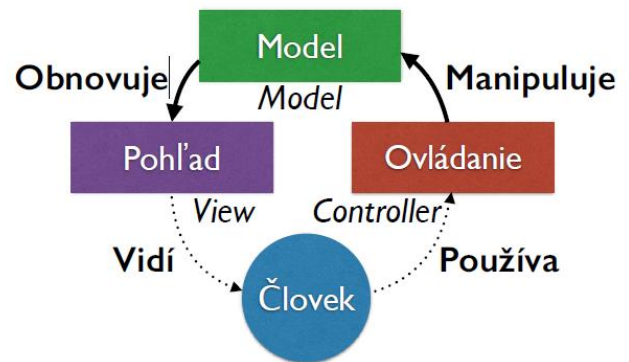


Vzor MVC

Model View Controller

- Základným problémom je určiť aká časť kódu je zodpovedná za aké operácie
- *Návrhové vzory* popisujú vysoko-úrovňovú organizáciu ktorá rieši bežné problémy
- V objektovo orientovanom programovaní je návrhový vzor základom z ktorého sa vychádza

MVC - Model, Pohľad, Ovládač



- **Model** - vnútorná reprezentácia dát aplikácie
 - Model je časťou ktorá vykonáva riešenie, všetku prácu - je modelom riešenia problému
 - Model by mal byť nezávislý od ostatných komponentov
- **View** - pohľad na model a jeho zobrazenie
 - **Pohľad** poskytuje náhľad na to čo **Model** vykonáva
 - **Pohľad** by nemal *zobrazovať* nič súvisiace s činnosťou **Ovládania**
- **Controller** - riadenie a spracovanie vstupov a zmien
 - Je skoro vždy možné **Ovládanie** a **Model** oddeliť
 - **Model** by sa nemal nikdy prispôbovať **Ovládaniu**
 - Zvyčajne sa tieto časti implementujú pomocou oddelených objektov alebo modulov

Rozhrania WIMPs

- Windows, Icons, Menus, Pointers
- Najčastejšie používaná paradigma v prostredí tvorby interaktívnych aplikácií
- Typické príklady: Aplikácie na MS Windows, Apple OSX alebo Linux Desktop aplikácie
- Implementácia zväčša daná systémom alebo GUI knižnicou

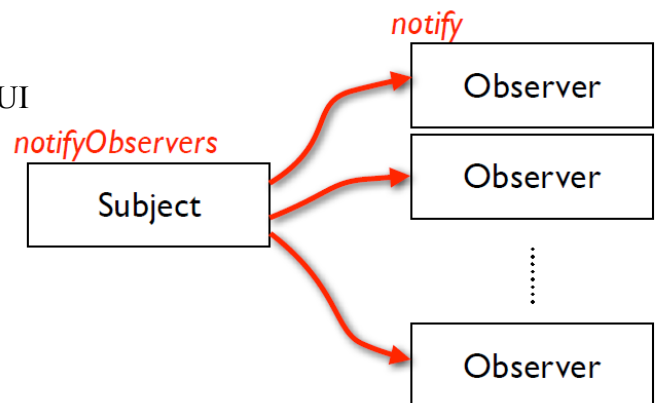
Interakcia vo WIMPs

- Používateľ by mal vedieť pracovať...
- *Len* s použitím myši
- *Len* s použitím klávesnice
- *Kombináciou* klávesnice a myši
- Práve **kombinovaný** prístup je efektívny
- Myš je neefektívna pri zadávaní textu
- Klávesnicou sa zle ukazuje na grafické prvky

Vzor Observer

- Návrhový vzor Observer, *Observer Pattern*
- Rieši problém šírenia správ a udalostí medzi objektami v aplikácii
- Kľúčová súčasť vzoru MVC a implementácie GUI aplikácií
- Súčasťou implementácie mnohých knižníc a systémov

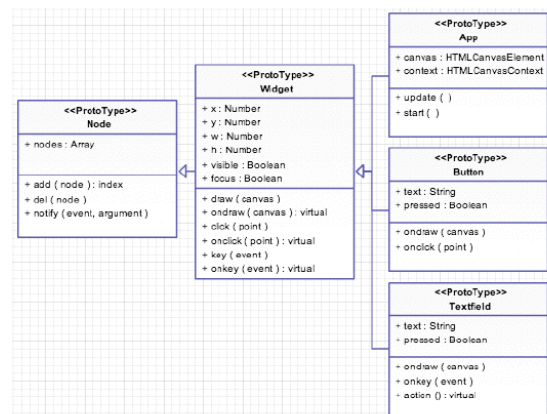
- Vzor pozostáva z dvoch objektov
- *Subject* je objekt ktorý je pozorovaný
- *Observer* je objekt ktorý pozoruje
- Pri výskyte udalosti informuje *Subject* všetky *Observer* objekty ktoré ho pozorujú o zmene
- *Observer* reaguje na zmenu



Widget

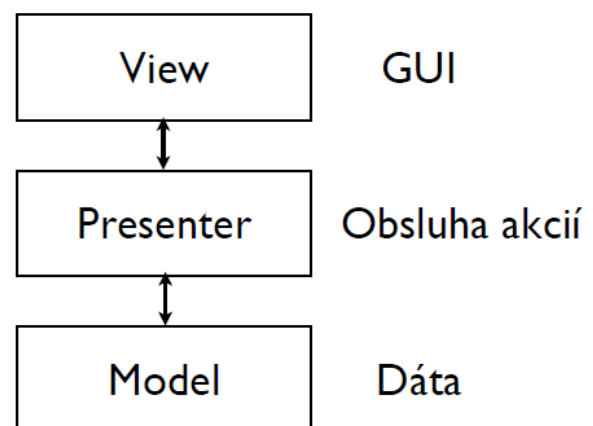
- Objekt reprezentujúci grafický element rozhrania aplikácie
- Typicky napríklad *button*, *textfield*, *menu* atd.
- Všetky widgety majú časť správania sa spoločnú
- Spracúvajú prichádzajúce správy (click, move, key)
- Organizované do stromov pre vytvorenie GUI

Dedenie vlastností



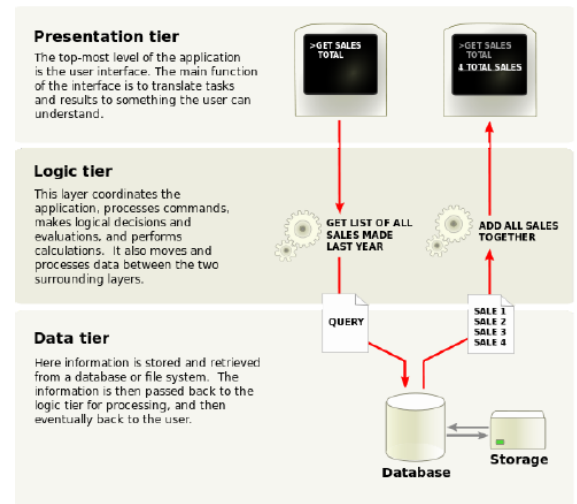
Model View Presenter

- Variácia myšlienky použitej v MVC
- Zamerané na moderné používateľské rozhrania a web aplikácie
- *View* predstavuje GUI aplikácie a jeho logika a implementácia je daná, napríklad HTML dokumentom
- Nie je nutné implementovať kód spracovania interakcie od používateľa, stačí obslúžiť len jeho akcie
- *Presenter* je zodpovedný za aplikačnú logiku a predstavuje jadro aplikácie, poskytuje používateľovi ovládanie cez *View*



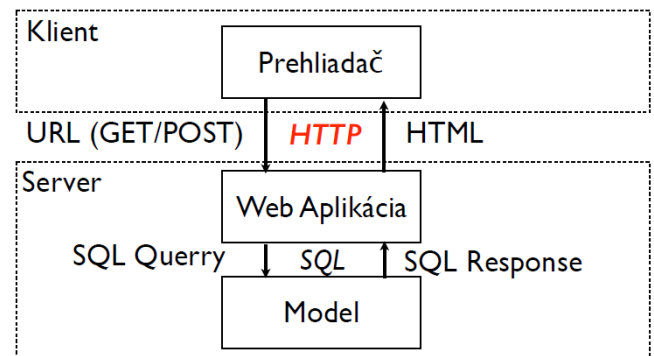
Trojvrstvová architektúra

- MVP je vzor pre návrh interakcie avšak možno ho zovšeobecniť pre návrh interaktívnych aplikácií
- Trojvrstvový návrh aplikácií typický pre Web a Klient-server aplikácie
- Typicky oddeľujeme prezentačnú vrstvu od logiky aplikácie a jej dát



Klient-server aplikácie

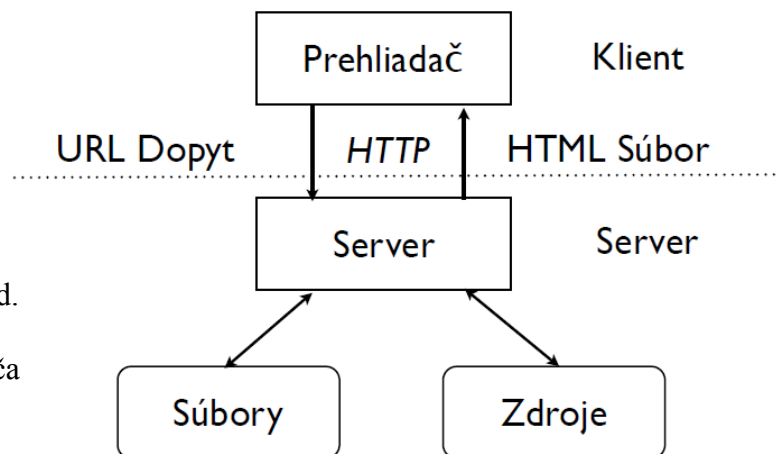
- Typická architektúra kde prehliadač vystupuje ako *View* pre aplikáciu bežiacu na strane servera
- Dáta sa do prehliadača šíria vo forme generovaného HTML ktorý vytvorí nový *View*
- Stránky generuje *Presenter* ktorý predstavuje jadro a logiku aplikácie
- Dáta sa ukladajú zväčša do SQL databáz



HTTP

- Hyper Text Transfer Protokol
- Definuje pravidlá komunikácie prehliadača so serverom
- Dopyt
- GET
- POST
- Odozva
- Stav
- Typické odpovede
- Dynamické HTML

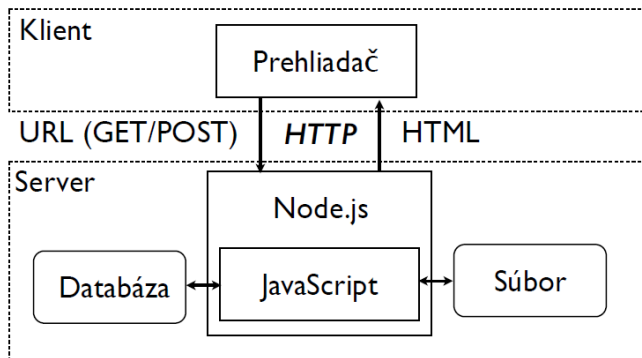
- Dopyt (Request)
 - Vyžiadanie stránky (pomocou URL)
 - Typicky HTML dokumenty, obrázky atd.
 - Obsahuje vstupy formulára
 - Obsahuje niektoré nastavenia prehliadača
- Odpoveď (Response)
 - Súbor (HTML alebo akýkoľvek iný súbor)
 - Obsahuje vlastnosti dokumentu
- Môže smerovať na iné dokumenty



- Metóda GET
- Na získanie (en. Getting) stránky
- URL vpísané ako adresa v prehliadači
- Linky v iných dokumentoch
- Môže zaslať obmedzené množstvo informácií
- Stránky možno pred-generovať (cache)

- Metóda POST
- Na zaslanie informácie (en. Posting)
- Môže obsahovať veľké množstvo dát
- Nie je možné použiť cache
- Neostávajú v histórii
- Nie je možné vytvoriť odkaz

Node.js aplikácie

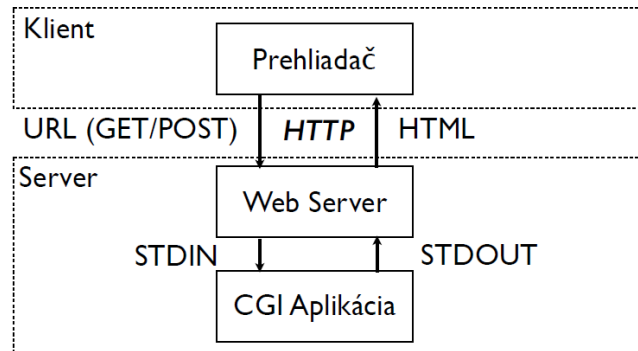


Moderné Web aplikácie

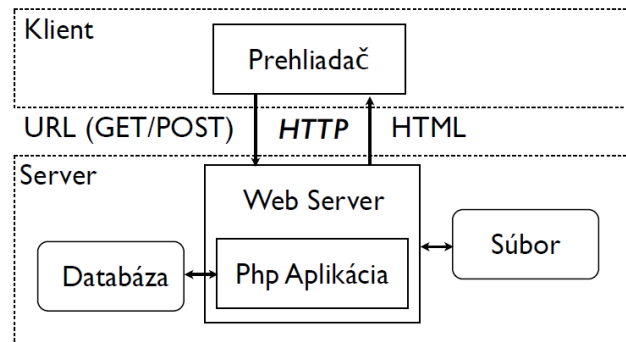
- Minimalizácia odozvy pri práci s aplikáciou
- Odstránenie nutnosti generovania stránky na strane servera po každej akcii
- Zmenšenie množstva prenesených dát
- Vyššia zložitosť, klient už nie je len *presenter* ale je integrálnou súčasťou aplikácie
- Ide o klient server architektúru ktorá stavia na Web technológiách

- *Frontend* - Časť aplikácie ktorá ma za úlohu poskytovať rozhranie pre používateľa.
- Typicky spája funkcionality *view* a *controller*
- *Backend* - Aplikačná časť ktorá obsahuje funkcionality očakávanú od časti *model*
- Komunikácia je realizovaná zasielaním HTTP požiadaviek na rozhranie, API backend-u

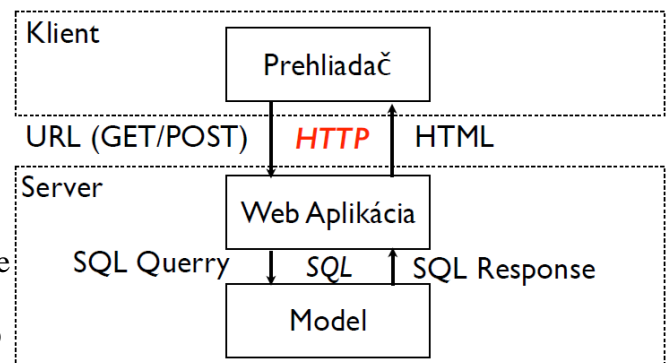
CGI aplikácie



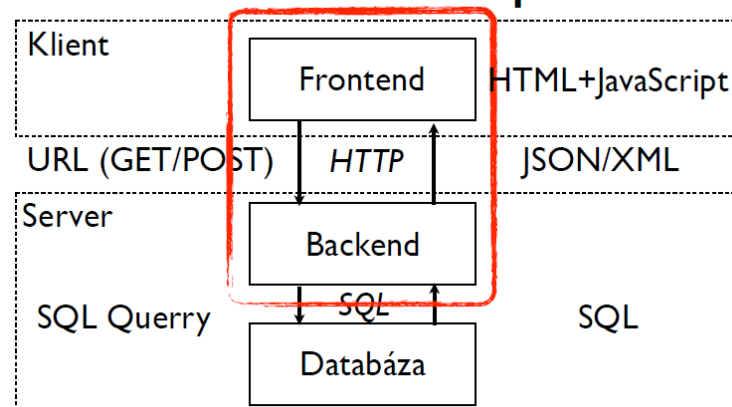
PHP aplikácie



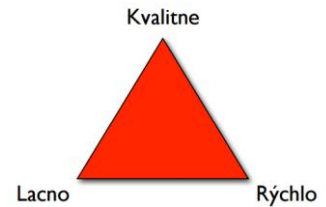
Klasické Web aplikácie



Moderné Web aplikácie



Problém



Stála Integrácia •

Continuous Integration (CI)

- Metodológia tvorby softvéru
- Automatizácia procesu testovania softvéru
- Zamerané na hlavné riziko
- Opravovanie chýb neskoro
- Vedľajšie riziká
- Problémy pri vzájomnej spolupráci vývojárov

GIT

- Vytvoril ho Linus Torvalds, 2005
- <http://www.git-scm.com/>
- Decentralizovaný systém pre manažment zmien súborov projektu
- Pôvodne bol určený ako nástroj pre údržbu Linux jadra
- Veľa zmien • Veľa autorov a prispievateľov
- Veľmi distribuovaná komunita • Konflikt z so systémom BitKeeper (komerčný systém)
- Základnou jeho črtou je že je distribuovaný
- Systém nemá žiaden centrálny server
- Každý adresár je kompletne úložisko (repozitár)
- Obsahuje kompletnú históriu projektu
- Viacero implementácií
- Pomerne jednoduchá implementácia na báze asociatívneho poľa