Računarstvo i informatika

Katedra za računarstvo
Elektronski fakultet u Nišu

Sistemi baza podataka

Razvoj aplikacija za rad sa bazama podataka

Letnji semestar 2015

Proces

- Osnovne aktivnosti
  - Planiranje (studija izvodljivosti)
    - Šta sistem treba da radi? (funkcionalnosti na visokom nivou)
    - Koliko će sistem da košta? Koje su dobre strane uvođenja sistema? Da li se uopšte isplati izrada aplikacije? Da li će sistem vratiti uložene resurse?
  - Analiza
    - Šta sistem treba da radi (Funkcionalnosti sistema sa mnogo više detalja: predlog delova UI, primeri izveštaja i slično)?
    - Analiza podataka (npr. Koje entitete treba čuvati u bazi podataka i koja su svojstva entiteta)

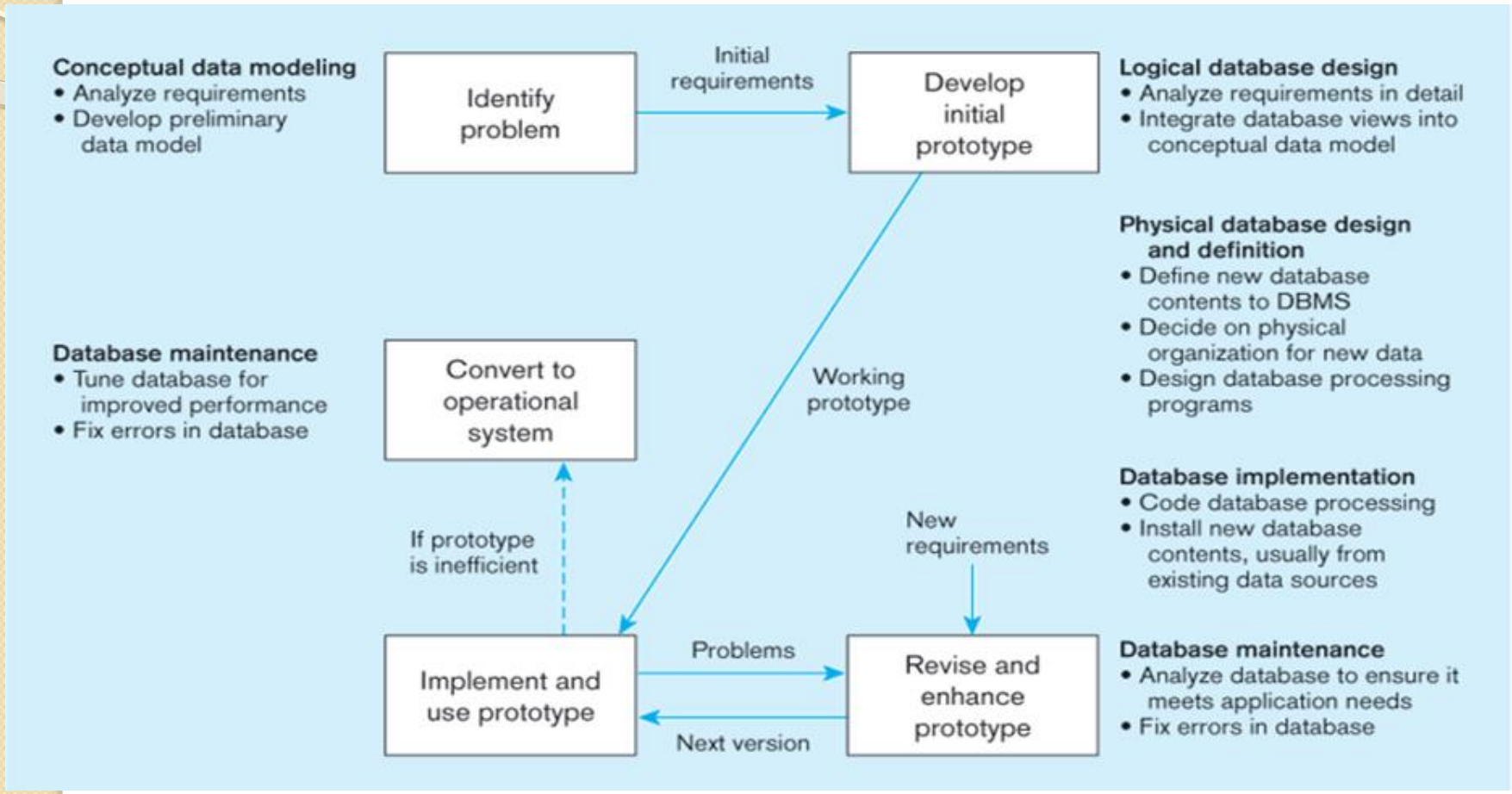
Proces

- Projektovanje
  - Projetovanje modela podataka
  - Projektovanje aplikacije
- Implementacija
  - Programiranje (pisanje koda na osnovu projekta sistema)
  - Testiranje
  - Isporuka sistema (korisnička dokumentacija, obuka korisnika, dokumentacija za upotrebu, obuka osoba zaduženih za održavanje sistema
- Održavanje sistema
  - Korisnici prijavljuju greške koje treba otkloniti
  - Korisnici zahtevaju nove funkcionalnosti (funkcionalnosti treba da prođu kroz mini ciklus razvoja)

Proces

- Različitim kombinovanjem osnovnih aktivnosti dobijaju se različiti modeli za razvoj aplikacija.
- System development lifecycle
  - Detaljan i dobro dokumentovan proces razvoja aplikacija
  - Sveobuhvatan proces koji zahteva dosta vremena
  - Dug razvojni ciklus
- Prototipovanje
  - Rapid application development (RAD)
  - Brz razvoj konceptualnog modela podataka
  - Kreiranje baze podataka tokom implementacije inicijalnog prototipa
  - Proces se ponavlja sa svakom novom verzijom prototipa

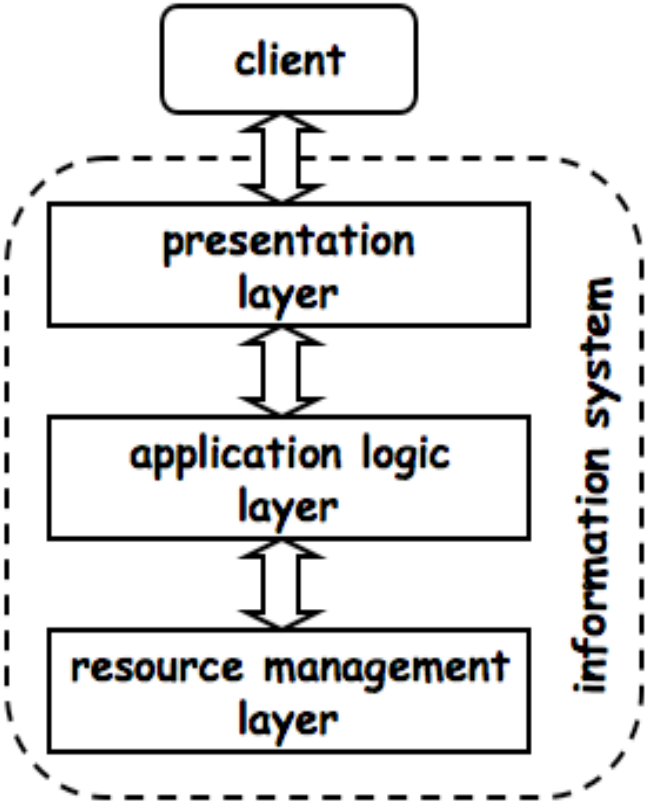
Proces


Conceptual data modeling
• Analyze requirements
• Develop preliminary data model
Identify problem
Initial requirements
Develop initial prototype
Logical database design
• Analyze requirements in detail
• Integrate database views into conceptual data model
Physical database design and definition
• Define new database contents to DBMS
• Decide on physical organization for new data
• Design database processing programs
Database maintenance
• Tune database for improved performance
• Fix errors in database
Convert to operational system
Working prototype
If prototype is inefficient
New requirements
Database implementation
• Code database processing
• Install new database contents, usually from existing data sources
Implement and use prototype
Problems
Next version
Revise and enhance prototype
Database maintenance
• Analyze database to ensure it meets application needs
• Fix errors in database

Proces

- Enterprise data model
  - Jedinstvena specifikacija visokog nivoa podataka koje će koristiti različite komponente sistema
  - Sveobuhvatna slika organizacije podataka na visokom nivou apstrakcije
  - Opis entiteta i relacija između entiteta koji su definisani poslovnim pravilma i procesima
  - Predstavljena korišćenjem nekog modela za konceptualno projektovanje (npr. EER model)
  - Kod kompleksnih sistema, koji se sastoje od većeg broja nezavisnih komponenti, jako je bitno kreirati ovaj model pre nego što se krene na projektovanje detaljnih modela

Arhitektura



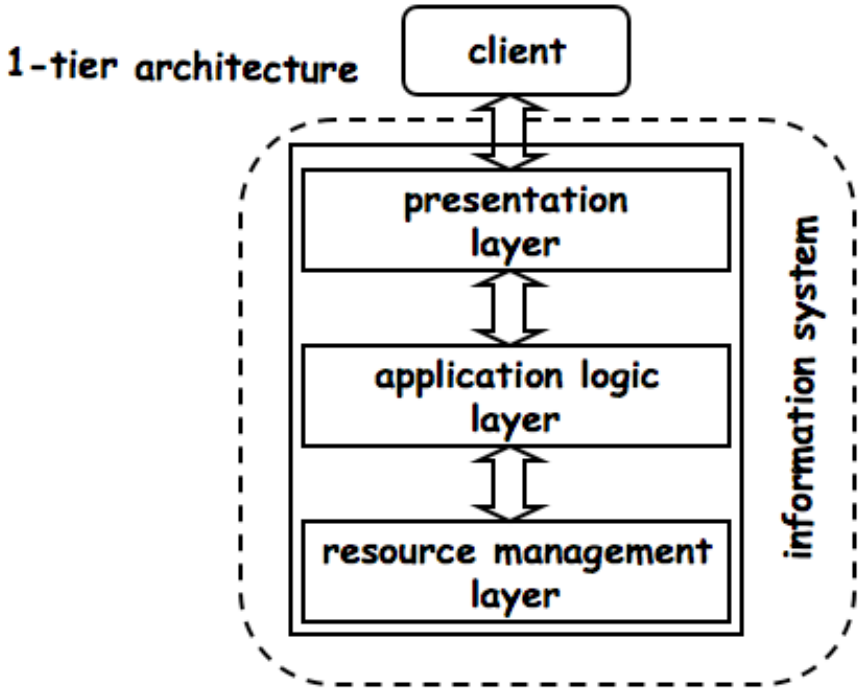
Klijent je bilo koji korisnik ili program koji želi da izvrši određenu operaciju nad sistemom.

Klijent interaguje sa sistemom korišćenjem **prezentacionog sloja** (korisnički interfejs)

Aplikativna logika definiše šta sistem radi. Nameće poslovna pravila i definiše poslovne procese.

Sloj za upravljanje resursima definiše organizaciju (skladištenje, indeksiranje i pribavljanje) podataka koji su neophodni kao podrška za aplikativnu logiku. Tipično se radi o bazi podataka.

Arhitektura



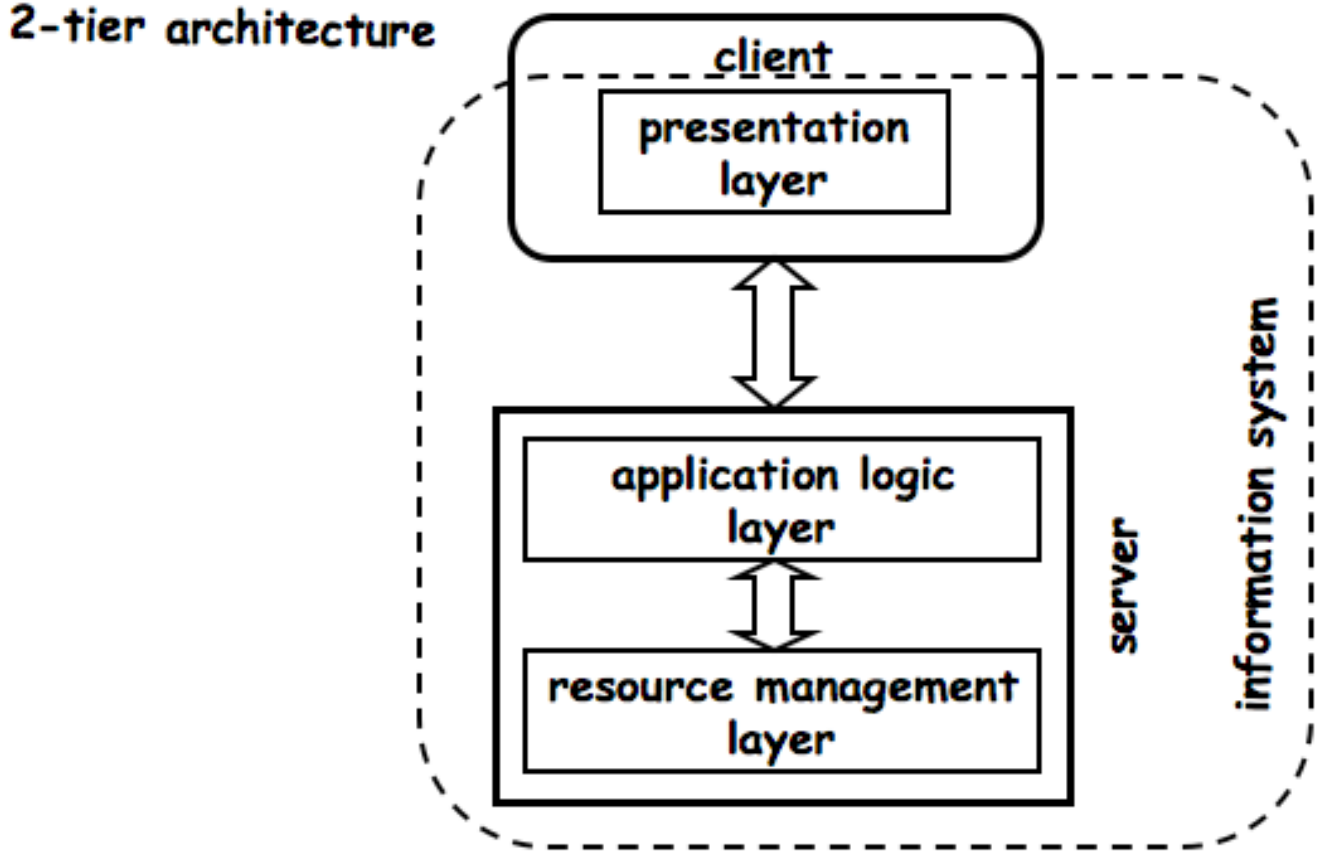
Jednoslojna arhitektura (1 tier)

Svi slojevi aplikacije su implementirani kao jedan monolitni entitet.

Kompletna aplikacija se nalazi na jednom računaru tj. na jednoj platformi.

Sve se dešava na jednom mestu.

Arhitektura



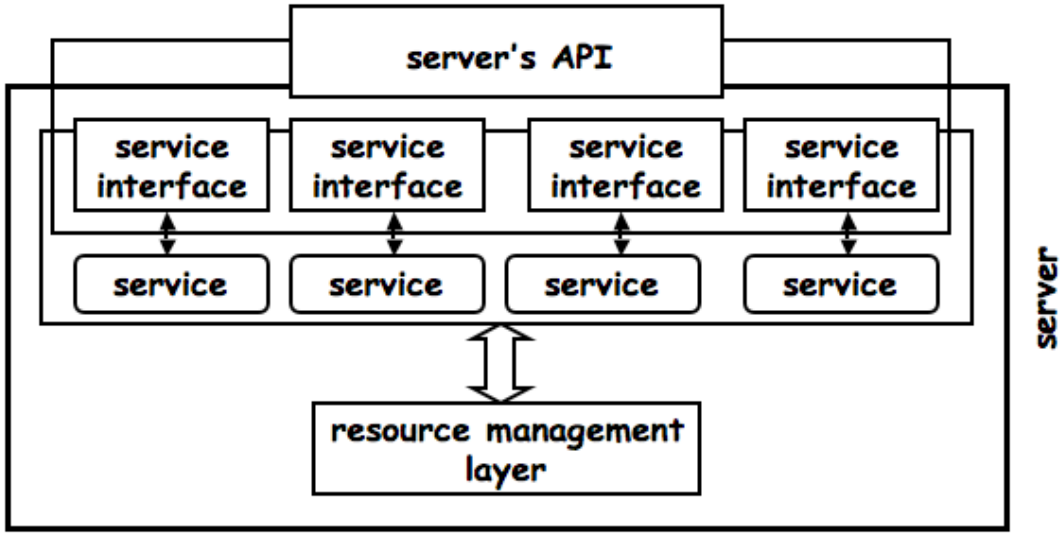
Arhitektura

- **Dvoslojna arhitektura (2 tier)**
  - Ovo je tradicionalna arhitektura i uključuje postojanje 2 računara koji učestvuju u izvršenju aplikacije.
  - Klijenti su međusobno nezavisni, pa za različite klijente može da postoji različiti prezentacioni sloj u zavisnosti od funkcionalnosti koje aplikacija nudi.
  - Klijenti su obično tanki (thin client), imaju vrlo ograničene mogućnosti.
  - Klijentima su se smatrali i prosti ulazni uređaji čiji zadatak je bio da samo prikažu klijentski interfejs koji se izvršava na udaljenoj platformi. Ovo su tzv. Vrlo tanki klijenti.
  - Na serveru se nalaze sve datoteke, baza podataka i komponente koje implementiraju aplikativnu logiku.
  - Baza podataka ima samo jednog klijenta: sloj aplikativne logike.

Arhitektura

- Vremenom kliejntski računari postaju sve moćniji.
- U cilju rasterećenja servera deo aplikativne logike se seli na klijentski računar.
- Samim tim klijenti dobijaju mogućnost da izvršavaju značajniji deo funkcionalnosti aplikacije pa se klijenti u tim situacijama smatraju debljim (fat client).
- Debljina klijenta varira od aplikacije do aplikacije i delom je odgovornost projektanata aplikacija.
- Ukoliko se zanemari prezentacioni sloj, na serverskoj strani se sve izvršava na jednom mestu poput sistema sa jednoslojnom arhitekturom.
- Definiše se pojam programskog interfejsa aplikacije (API) – interfejs koji omogućava da se elementi aplikativne logike pozovu spolja.

Arhitektura

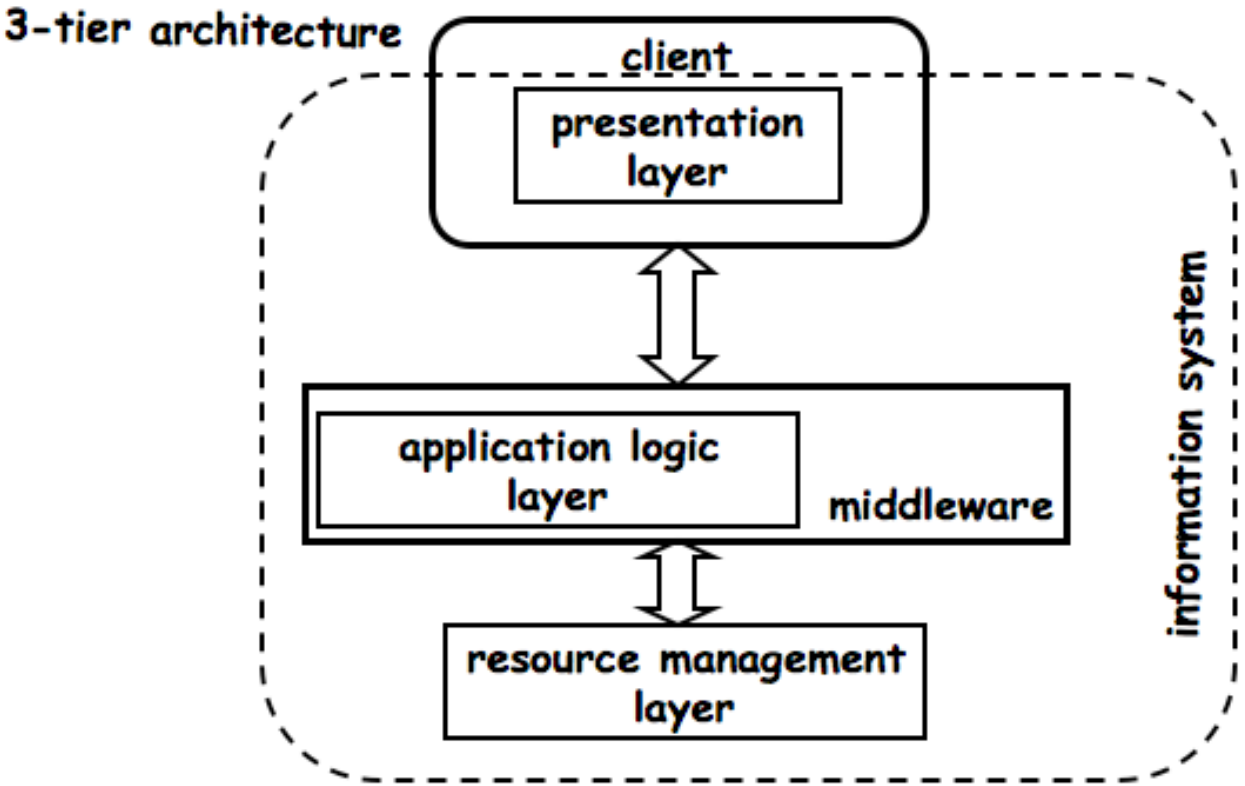


Dvoslojna klijent/server arhitektura uvodi pojam servisa. Klijenti koriste servise koje server implementira.

Takođe se uvodi pojam interfejsa servisa. Interfejs servisa definiše kako klijent može koristiti servis.

Interfejsi svih servisa koje server implementira predstavlja serverski API.

Arhitektura

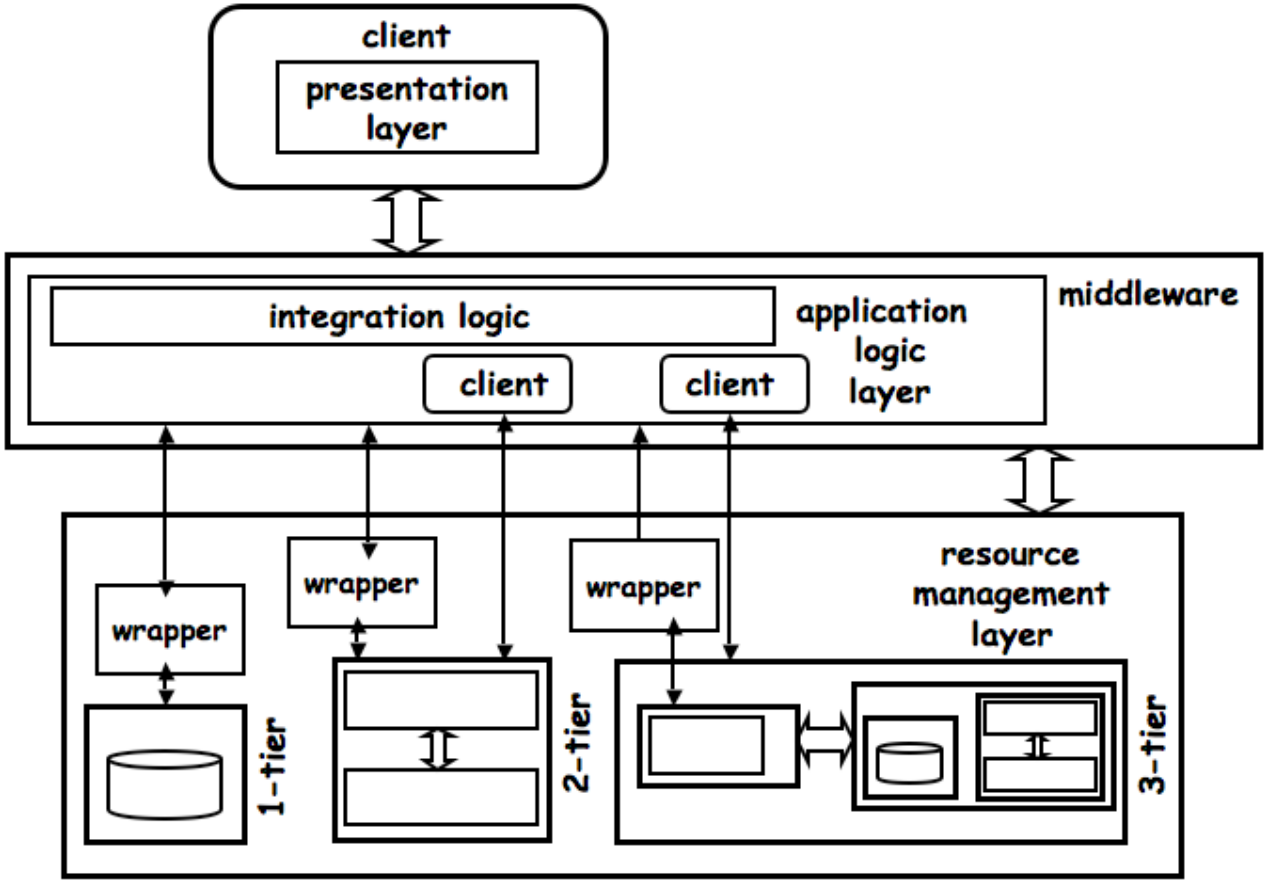


Arhitektura

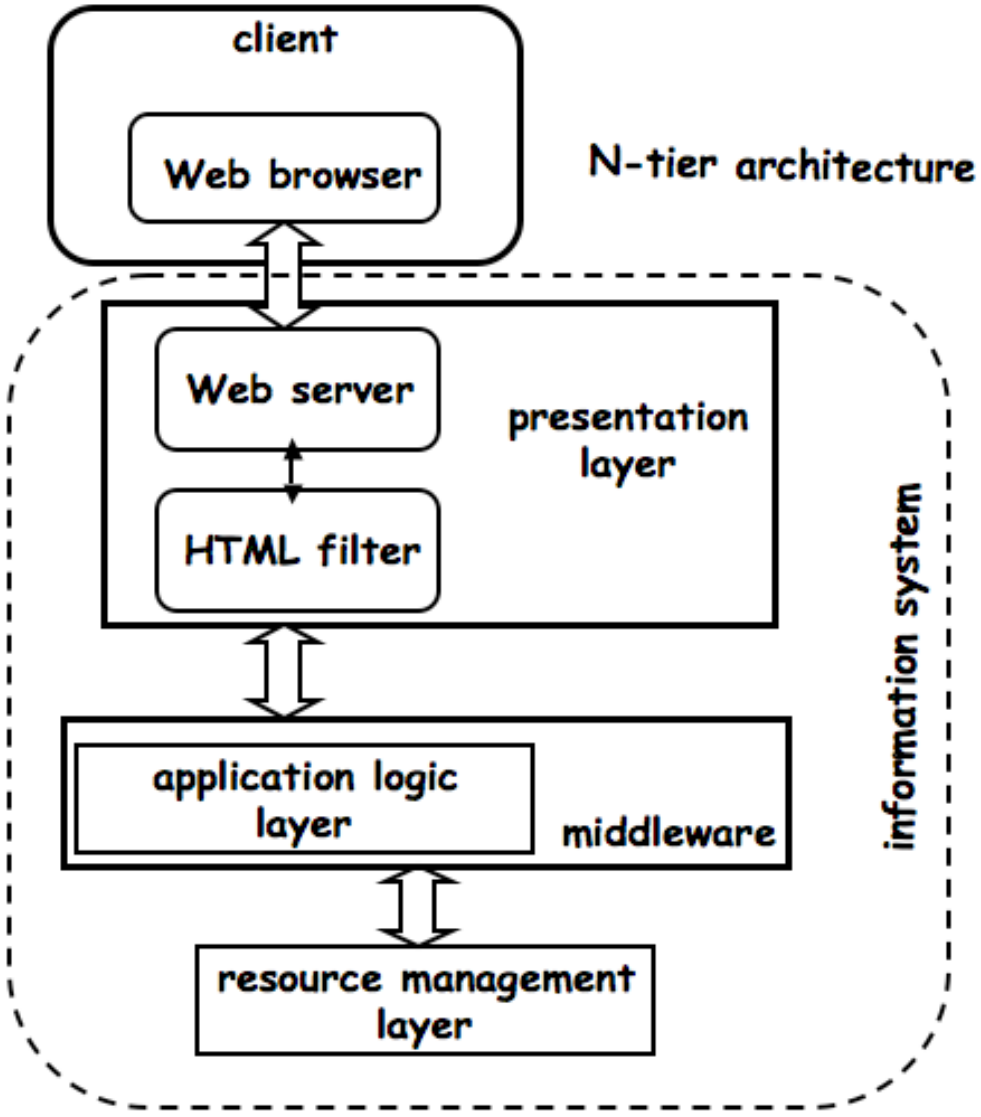
- **Troslojna arhitektura (3 tier)**
  - Dvoslojne arhitekture su bile popularne do trenutka kada je postalo jasno da je opterećenje na serverskoj strani moguće dodatno raspodeliti odnosno distribuirati.
  - Osnovni problem dvoslojne arhitekture je jaka sprega između aplikativne logike i izvora podataka koji aplikacija koristi.
  - Kod troslojni sistema tri sloja aplikacije su potpuno odvojena i u vićeni slučajeva distribuirana na različitim platformama.
  - Srednji sloj obuhvata kompletnu aplikativnu logiku i naziva se middleware ili aplikativni server.
  - Bolja raspodela opterećenja i bolja skalabilnost sistema.

Arhitektura

- Integracija sistema



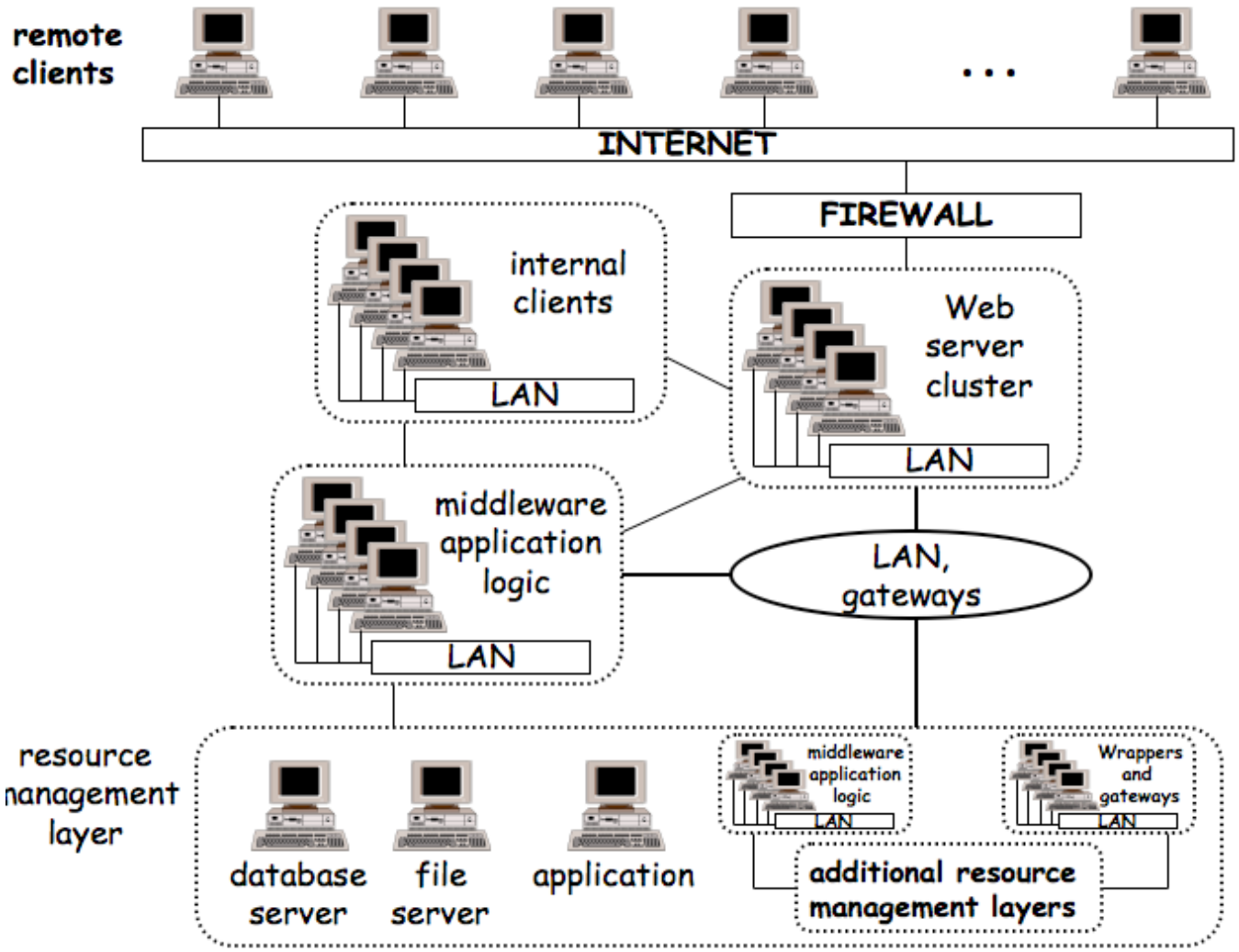
Arhitektura



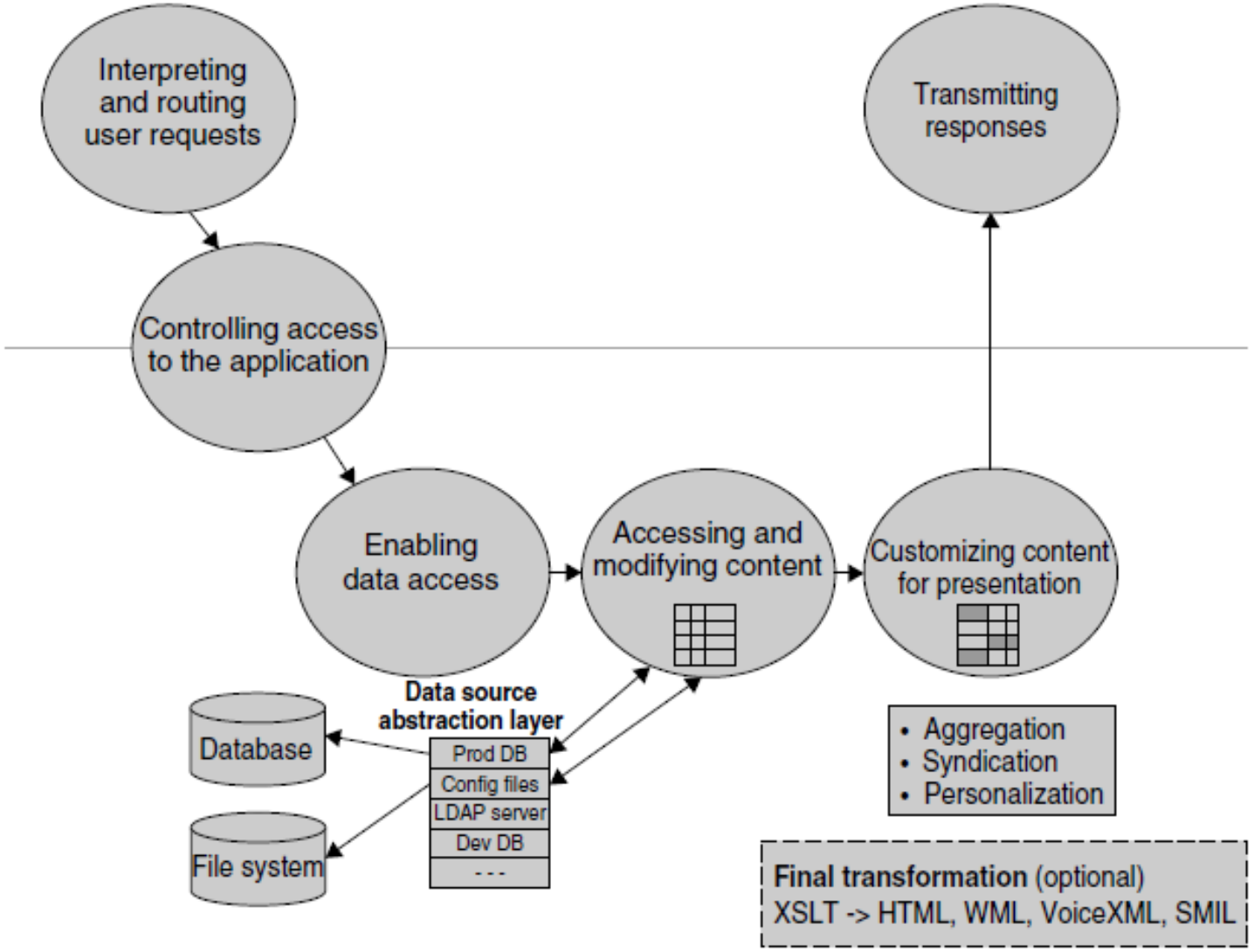
Arhitektura

- **Višeslojna (N-tier) arhitektura**
  - Web serveri se često ponašaju kao klijenti u situacijama kada koriste usluge drugih servera, uključujući i servere baza podataka
  - Višeslojne aplikacije poseduju veći broj slojeva koji se mogu ponašati i kao klijenti i kao serveri.
  - Svaki skup susednih slojeva predstavlja klijent-server uparivanje.
  - U višeslojnim aplikacijama svaki sloj predstavlja aplikacioni sloj (browser, Web server, aplikacioni server, baza podataka...)
  - Poslednji sloj u arhitekturi je Web klijent (u većini slučajeva Web čitač ili inteligentni agent) koji inicira obradu podataka slanjem podataka Web serveru.

Arhitektura



Obrada podataka



Obrada podataka

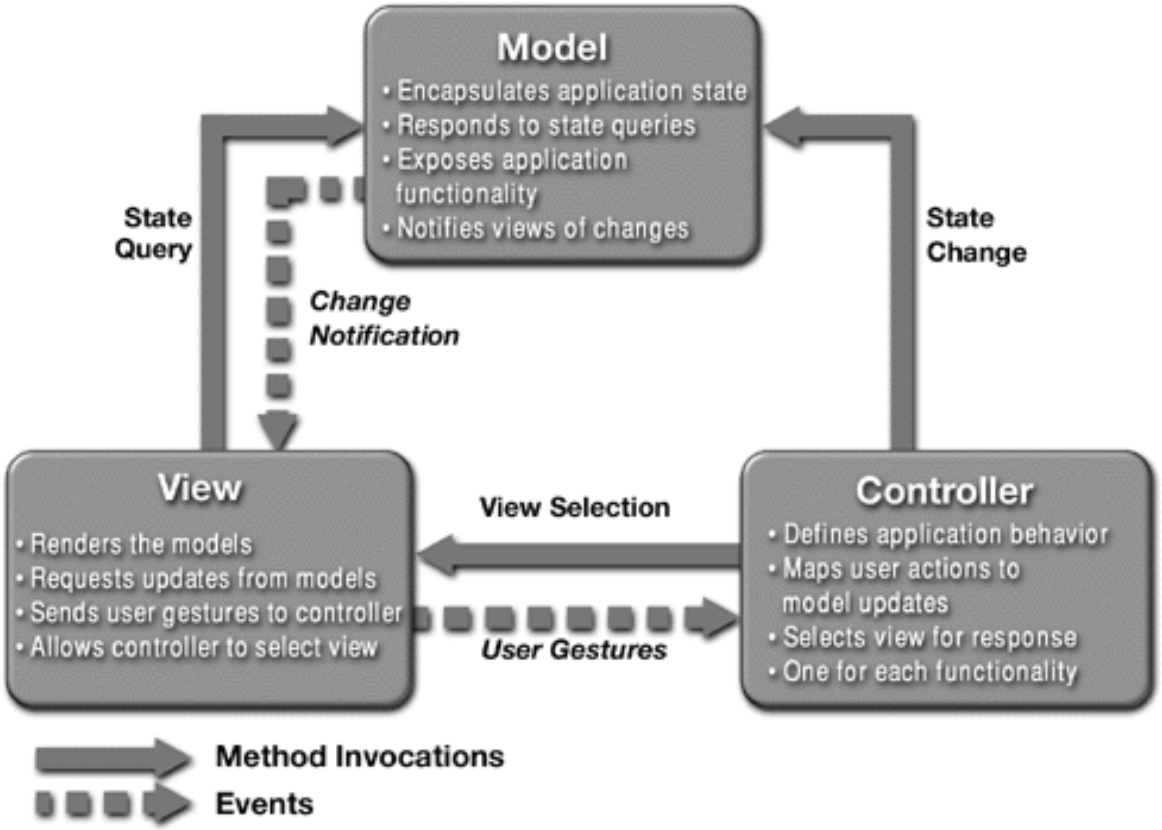
- **Interpretiranje i rutiranje zahteva korisnika**: Web server određuje koje akcije bi trebalo sprovesti kako bi se procesirao zahtev korisnika.
- **Kontrola pristupa aplikaciji**: U većini situacija neophodni je posedovati više od proste mogućnosti autentifikacije korisnika. Često se funkcionalnosti aplikacije dele po grupama korisnika.
- **Kontrola pristupa podacima**: deljenje pristupa informacijama u zavisnosti od nivoa privilegija autorizovanog korisnika.
- **Modifikacija podataka**: Pored pristupa i prikaza podataka, aplikacije su često u stanju da modifikuju podatke. Modifikacija podataka može biti posledica akcija korisnika ili se može desiti automatski na osnovu aplikacione logike.
- **Prilagođenje odgovora**: Odgovor na zahtev korisnika je neophodno prilagoditi kako bi njegov konačni prikaz bio prilagođen potrebama i očekivanjima korisnika.

Razdvajanje slojeva

- Osnovu svake aplikacije čine podaci, logika za njihovu obradu i način njihovog prikaza.
- Ove tri komponente često predstavljaju poprilično nezavisne aspekte razvoja aplikacije.
- Posmatrano iz ugla korišćenja podataka, nebitan je izvor iz koga su podaci pribavljeni (baza podataka, fajl sistem, email itd.)
- Veoma je bitno izabrati otvoreni model za predstavljanje i obradu podatka. Time se omogućava vizuelizacija informacija na različite načine tj. kreiranje različite poglede na isti model podataka.
- Za razdvajanje slojeva mogu se koristiti različiti projektni obrasci.

Razdvajanje slojeva

- **Model-View-Controller (MVC)**

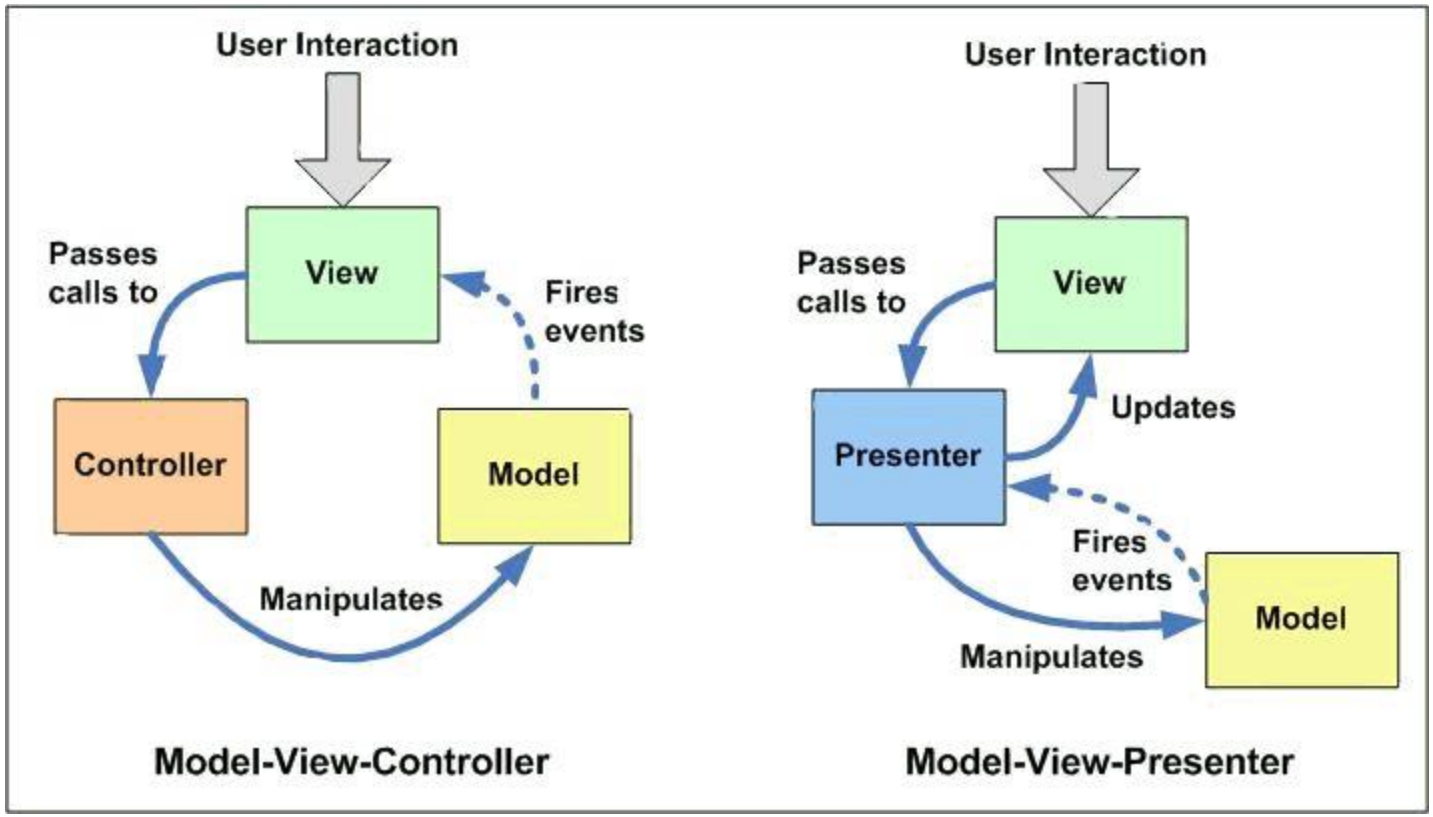


Razdvajanje slojeva

- **Model-View-Controller (MVC)**
  - Kontroler može da šalje komande modelu kako bi ažurirao njegovo stanje (npr. Izmena dokumenta). Kontroler šalje komande pridruženom pogledu kako bi promenio prezentaciju modela (npr. skrolovanje dokumenta).
  - Model šalje notifikacije pridruženim kontrolerima i pogledima kada promeni svoje stanje. Ove notifikacije omogućavaju pogledima da generišu izmenjenu reprezentaciju modela a kontrolerima da izmene dostupni skup komandi. U nekim slučajevima implementacija MVC obrasca je pasivna i pogleda i kontroleri moraju da prozivaju model kako bi detektovali izmene u njegovom stanju.
  - Pogled zahteva informacije od modela na osnovu kojih generiše reprezentaciju za potrebe korisnika.

Razdvajanje slojeva

- **Model-View-Presenter (MVP)**

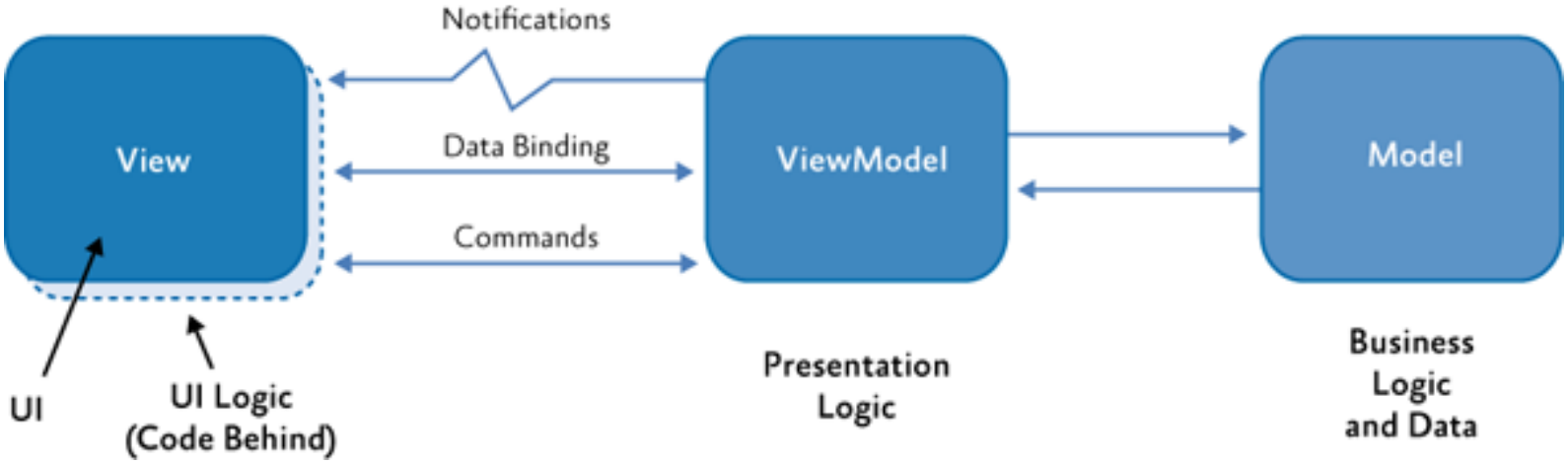


Razdvajanje slojeva

- **Model-View-Presenter**
  - Model predstavlja interfejs koji definiše podatke koji će biti prikazani ili na drugi način reaguje na akcije korisnika.
  - Pogled predstavlja pasivni interfejs koji prikazuje podatke (pribavljene od modela) i rutira korisničke komande (događaje) prezenteru radi dalje obrade.
  - Prezenter predstavlja sponu između pogleda i modela. Na osnovu primljenig događaja, pribavlja podatke iz modela i formatira za potrebe prikaza u pogledu.

Razdvajanje slojeva

- **Model-View-ViewModel (MVVM)**



Razdvajanje slojeva

- **Model-View-ViewModel (MVVM)**
  - Model je skup klasa koje predstavljaju objektnu reprezentaciju podataka pribavljenih iz izvora podataka (npr. baza podataka ili servis)
  - Pogled (View) je komponenta sistema odgovorna za prezentaciju informacija.
  - ViewModel priprema podatke pribavljene iz modela za prikaz u pogledu. ViewModel se često opisuje kao reprezentacija stanja podataka u modelu.