

# **Távközlési informatika**

## **Kliens-szerver architektúrák**

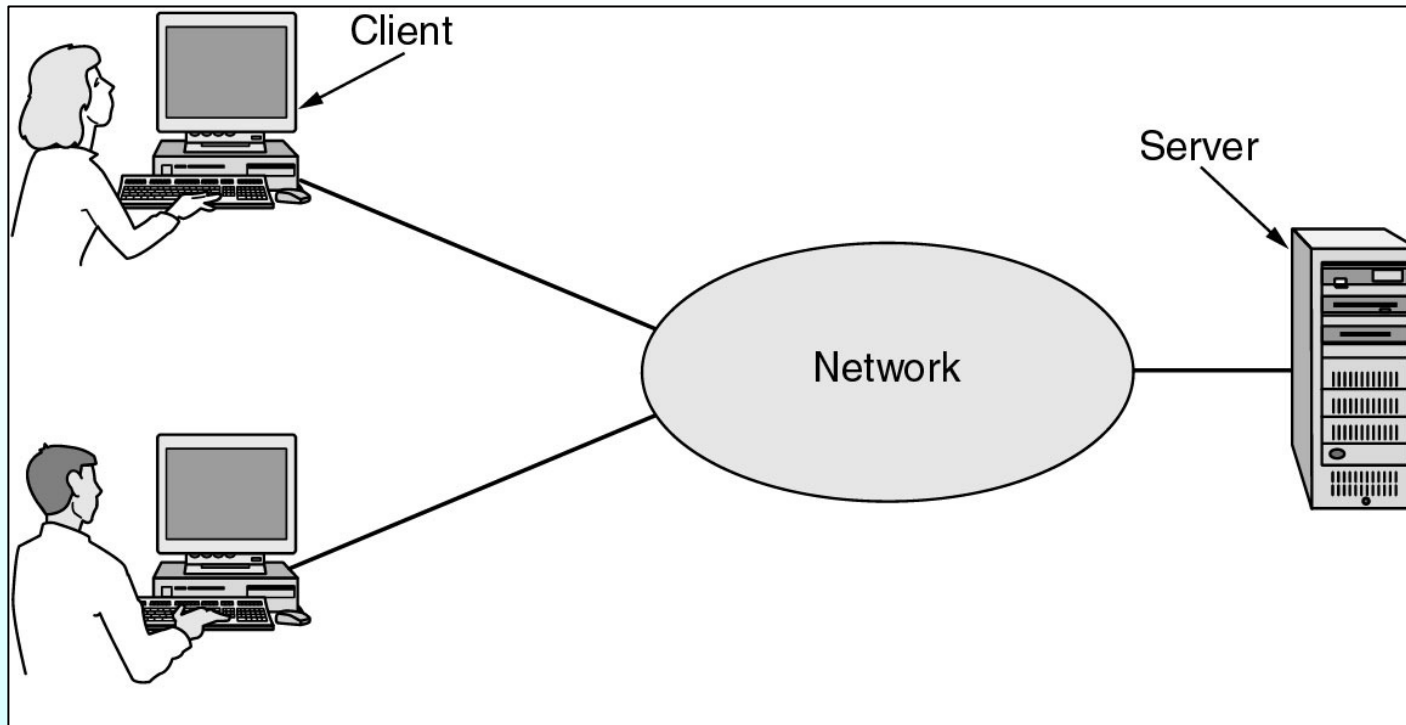
Dr. Beinschróth József

# Számítógép hálózatok osztályozása

**Többféle osztályozás lehetséges:**

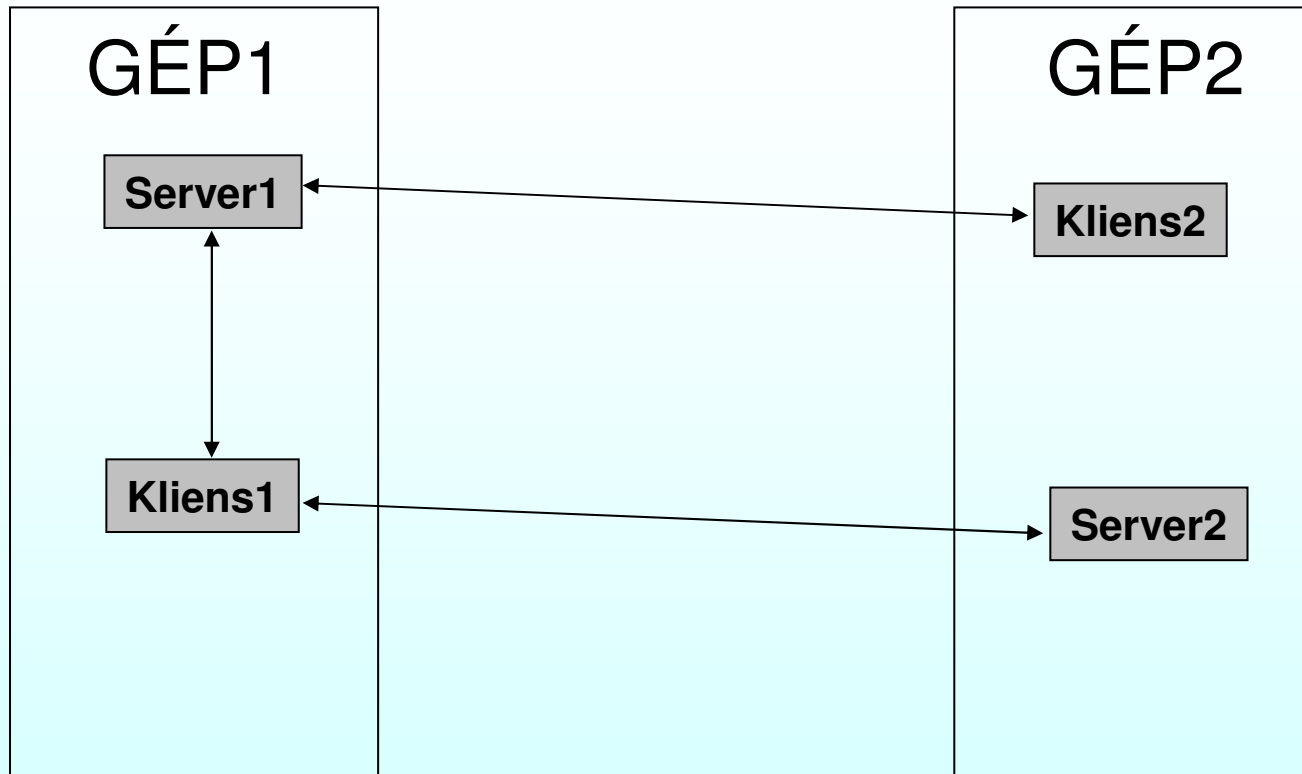
- **Kiterjedés szerint: LAN, MAN, WAN, (PAN)**
- **Hierarchia szerint:**
  - **peer-to-peer hálózatok (egyenrangú gépek, mindegyikük nyújthat és vehet igénybe szolgáltatásokat)**
  - **kliens-szerver (c/s) hálózatok (a szerverek által nyújtott szolgáltatásokat a kliensek igénybe veszik) – a TCP/IP alkalmazások tipikusan a c/s modell szerint működnek**
  - **hibrid hálózat (egyszerre vannak jelen az egyenrangú valamint a c/s hálózatokat alkotó technológiák)**

# Fogalmak: kliens, szerver



- **Szerver**
  - Kereskedelmi kategória, gép, process
  - PROCESS: Más processeknek szolgáltatást nyújt
- **Kliens**
  - Más processek (szerverek) szolgáltatását igénybe veszi
  - A kapcsolatot mindig a kliens kezdeményezi!

# Kliens-szerver (ügyfél-kiszolgáló) modell



- A kliens (front-end) valamilyen kéréssel fordul a szerver (back-end) felé, amely a kért szolgáltatást nyújtja
- A kommunikációt mindig a kliens kezdeményezi, sohasem a szerver
- Az kliens egy felhasználói program pl. levelezőprogram, a böngésző, stb., a szerver pl. web szerver

# Kliens-szerver modell - jellemzők

- A kliensen dolgozó felhasználó úgy érzi, mintha a feladat csak az ő számítógépe erőforrásai segítségével hajtodna végre, pedig a valóságban a futás igénybe veszi mind a kliens, mind a szerver erőforrásait és a hálózatot
- Olyan architektúra, amelyen a kommunikációban résztvevő két fél nem egyenrangú módon vesz részt, hanem dedikált szerepeket (szerver illetve kliens) töltenek be.
- Terheléselosztás valósul meg a kliens és szerver gép között (általában egy szerver több klienst is kiszolgál)
- Ha minden felhasználó bejelentkezne a szerverre, és az ott található programot használná, akkor a szerveren hatalmas erőforrás igény jelentkezne (a program egyszerre sok példányban futna a szerveren)
- A szerver gépen az erőforrás-igény csökken a feldolgozás egy része a kliens gépen történik. A szerverrel a kapcsolat csak akkor épül ki, ha adatcserére van szükség
- A kliens és a szerver egymástól tetszőleges távolságban is lehet
- (A kliens/szerver modell programozási technológiákban a főprogram-alprogram viszonynak feleltethető meg.)

# Kapcsolódási változatok

- Összeköttetés alapú kapcsolat
  - Összeköttetés-létesítési fázis (kapcsolatfelvétel)
  - Adatátviteli fázis
  - Összeköttetés-lebontási fázis (kapcsolat lebontása, miután nem akarunk több adatot küldeni vagy fogadni)
- Összeköttetés mentes kapcsolat
  - Az összeköttetés-mentes kapcsolatnál nincs egy összeköttetés-felépítési fázis, hanem bármelyik fél bármikor, bárkinek küldhet adatcsomagokat

# Összeköttetés alapú szerver

1. A szerver lefoglal egy kommunikációs végpontot (egy TCP-portot), hozzárendeli a megfelelő címet (általában a szolgáltatás megfelelő jól ismert TCP-portjának az azonosítóját –pl. smtp: 25-ös port)
2. A szerver várakozik egy kliens alkalmazás összeköttetés-létesítési kérelmére
3. A szerver felveszi a kapcsolatot a szolgáltatásait igénybe venni szándékozó kliens alkalmazással (felépíti a következő klienssel az összeköttetést)
4. Ebben a lépésben történhet a kliens alkalmazás kiszolgálása,
5. A klienssel felépített összeköttetés lebontása.  
(A szerver a második pontban folytatja: egy új kliensre várakozik. )

# Összeköttetés alapú kliens

1. Az alkalmazás lefoglal egy mások által nem használt (szabad) kommunikációs végpontot (pl. Java alkalmazások esetében TCP-portot). (Ez nem a jól ismert port!)
2. Az alkalmazás létrehoz egy hálózati összeköttetést a saját és a szerver kommunikációs végpontja között (összekapcsolódik a szerverrel)
3. Az alkalmazás adatokat küld a szervernek, és fogadja a szervertől érkező adatokat
4. A szolgáltatás igénybevétele után az alkalmazás lebontja a szerverrel felépített összeköttetést, és felszabadítja az első pontban lefoglalt kommunikációs végpontot (lehetővé téve az operációs rendszernek ennek az erőforrásnak más célokra használását).



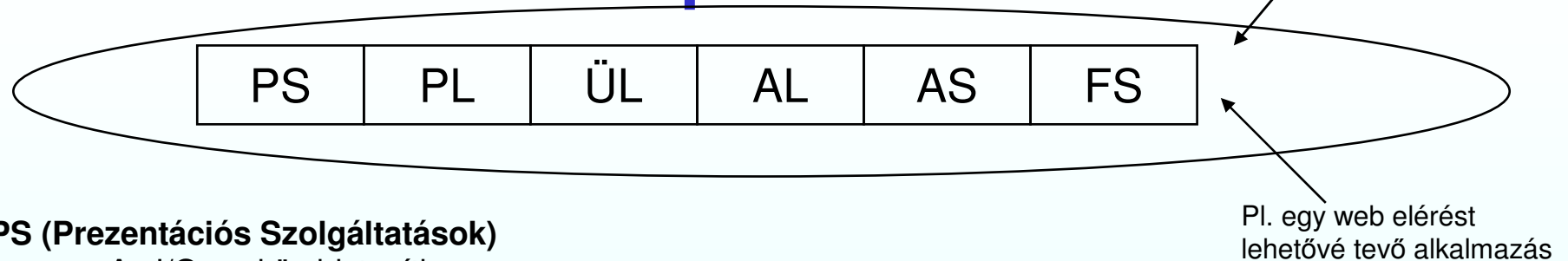
# Összeköttetés mentes szerver és kliens

- Tipikusan UDP fölött
- Szerver:
  - Fogadja az UDP portra érkező csomagot és csomagot tesz ki az UDP portra
- Kliens:
  - Fogadja az UDP portra érkező csomagot és csomagot tesz ki az UDP portra

# Kliens-szerver architektúrák (C/S)

- Az alkalmazások részekre bonthatók. Az egyes részek bizonyos rétegekbe (tíre) csoportosíthatók. (Az egyes részek között nem mindig teljesen egyértelmű a határ!)
- Csoportosítás: Mely részek kerülnek a szerverre ill. a kliensre?
- Kategóriák
  - Nem igazi C/S architektúrák
    - Mainframe stílusú adatfeldolgozás
    - Fájlszerver alapú adatfeldolgozás
  - Valódi C/S architektúrák (Az alkalmazásnak legalább egy meghatározott része vagy annál több helyezkedik el a kliensen)
    - Kétrétegű kliens/szerver (C/S) architektúrák (2 tíre)
    - Háromrétegű C/S architektúra (3 tíre)

# Alkalmazások felépítése



## •PS (Prezentációs Szolgáltatások)

- Az I/O eszköz biztosítja
- Elfogadja az inputot a felhasználótól
- Megjeleníti a Prezentációs Logikától kapott outputot

## •PL (Prezentációs Logika)

- Vezérli az ember-számítógép interakciót. (Mi történjen, ha a felhasználó kiválaszt egy menütételt vagy lenyom egy billentyűt?)

## •ÜL (Üzleti vagy alkalmazási Logika)

- Az üzleti követelményeket kielégítő döntések és számítások (pl. ÁFA számítás).

## •AL (Adat Logika)

- Az adatbázison végrehajtott műveletek az ÜL számításai számára (pl. adatbázis lekérdezések, különböző SQL utasítások).

## •AS (Adat Szolgáltatások)

- Az adatbáziskezelő tevékenysége, amely az AL igényeit elégíti ki
  - adatmanipuláció az adatbázisban
  - tranzakciókezelés
  - hozzáférési jogosultságok ellenőrzése
  - stb.

## •FS (Fájl Szolgáltatások)

- A bitek kiolvasása a lemezen lévő fájllokból, melyekből az adatbáziskezelő értelmezhető adatot formál
- (Rendszerint operációs rendszer szintű szolgáltatás)

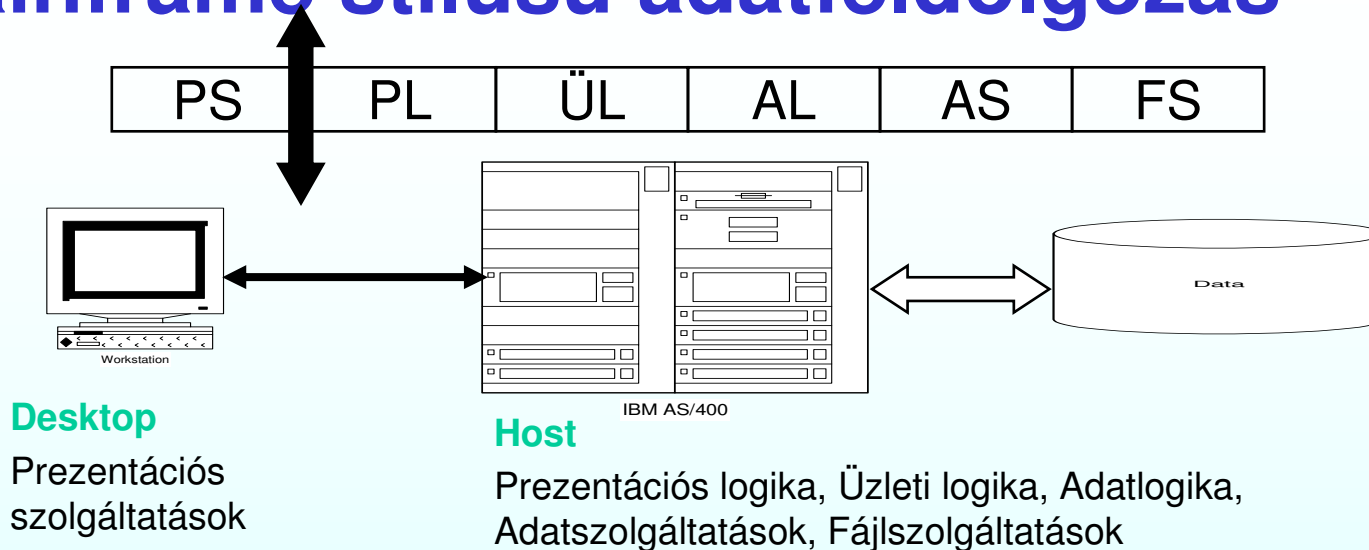
# Alkalmazások felépítése

**Mely részek kerüljenek a kliensre ill. a szerverre?**

**Alapvetően meghatározza a kliens, a szerver és a hálózat terhelését!**

- Mainframe stílusú adatfeldolgozás
- Fájlszerver alapú feldolgozás
- Távoli adat C/S Távoli prezentáció C/S
- Megosztott logikájú C/S
- Háromrétegű C/S
- ....

# Mainframe stílusú adatfeldolgozás

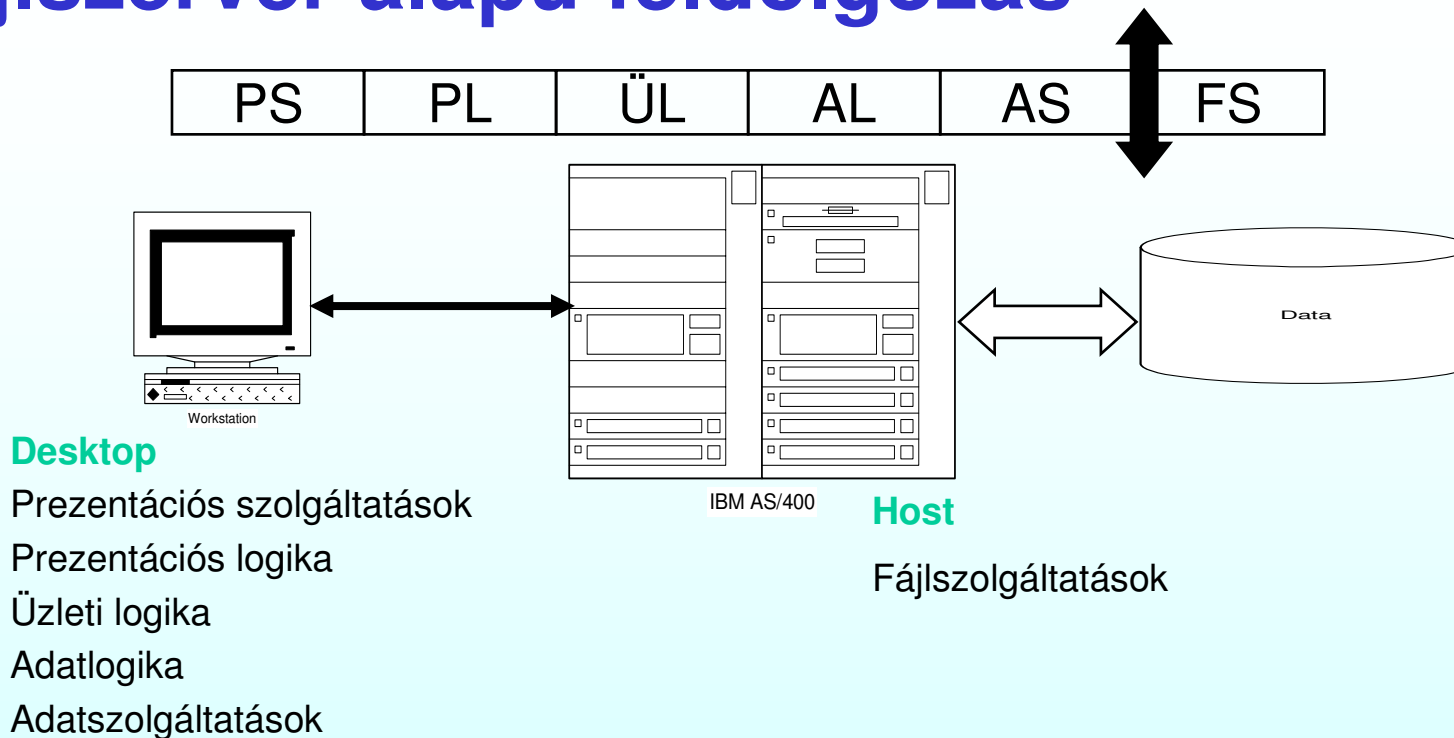


- **Centralizált, többfelhasználós feldolgozás**
- **Terminál interakció: klasszikus, többfelhasználós, interaktív rendszerek**
  - A terminál csupán a billentyűlévételeket továbbítja a mainframe-nek és megjeleníti mindazt, amit tőle kap
- **A terminál csak megjeleníti a karaktereket, minden egyebet a mainframe végez**
  - Felhasználói lekérdezések megválaszolása
  - Üzleti logika érvényesítése
  - Adatmanipuláció
  - Szokásos elnevezés: vékony kliens
- **A mainframe stílusú feldolgozáshoz nem szükséges klasszikus mainframe. Elegendő lehet egy megfelelően kiépített PC pl. Unix operációs rendszerrel és terminál(ok)kal.**

# Mainframe stílusú adatfeldolgozás

- Fő problémák
  - A felhasználói interfész karakter típusú terminálinterakcióra korlátozódik – GUI megvalósítása problematikus
  - A mainframe teljesítménye kritikus lehet - igen könnyen leterheltté válhat a felhasználók számának növekedtével
  - A hálózaton kapacitása ugyancsak kritikus lehet - nemcsak az adatok, hanem a megjelenítést vezérlő információk is a hálózaton továbbítódnak

# Fájlserver alapú feldolgozás



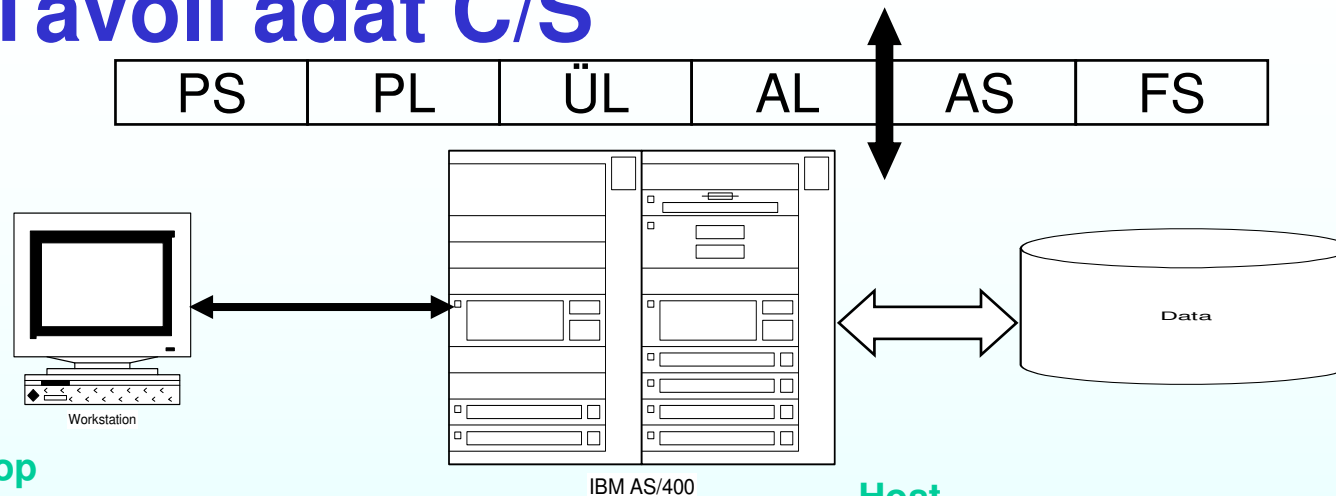
- Elsősorban fájlok és egyéb erőforrások (pl. nyomtató) megosztására jött létre hálózatba kötött PC-vel rendelkező felhasználók között
- Nem biztosítanak jó adatmegosztást - az adatmegosztás igen megterheli a hálózatot
- A fájlok kezelésén kívül minden feladat a desktopra hárul
  - Megjelenítés
  - Üzleti logika
  - Adatbázis funkciók
  - Szokásos elnevezés: vastag kliens
- A fájlserver csak előveszi a fájlokat és továbbítja azokat a hálózaton keresztül desktopnak

# Fájlszerver alapú feldolgozás

- Előnyök: A mainframe megoldásra jellemző problémák itt nem lépnek fel
  - A PS és a PL a desktopon van, ezért rugalmas GUI (grafikus felhasználói interfész) készíthető
  - A fájlserver csak fájlokkal foglalkozik, egyéb leterheltsége nincs.
  - Minden egyes új PC bekapcsolása a hálózatba új processzor erőforrást is jelent
- Hátrányok
  - Nagyteljesítményű desktopokra van szükség - op. rendszer és a teljes alkalmazás adatbázis-kezelés rajta fut
  - A hálózat igen könnyen leterhelődhet (ha a desktopnak egyetlen egy rekordra lenne szüksége, a fájlserver akkor is elküldi az egész fájlt az összes rekordjával)



# Távoli adat C/S



## Desktop

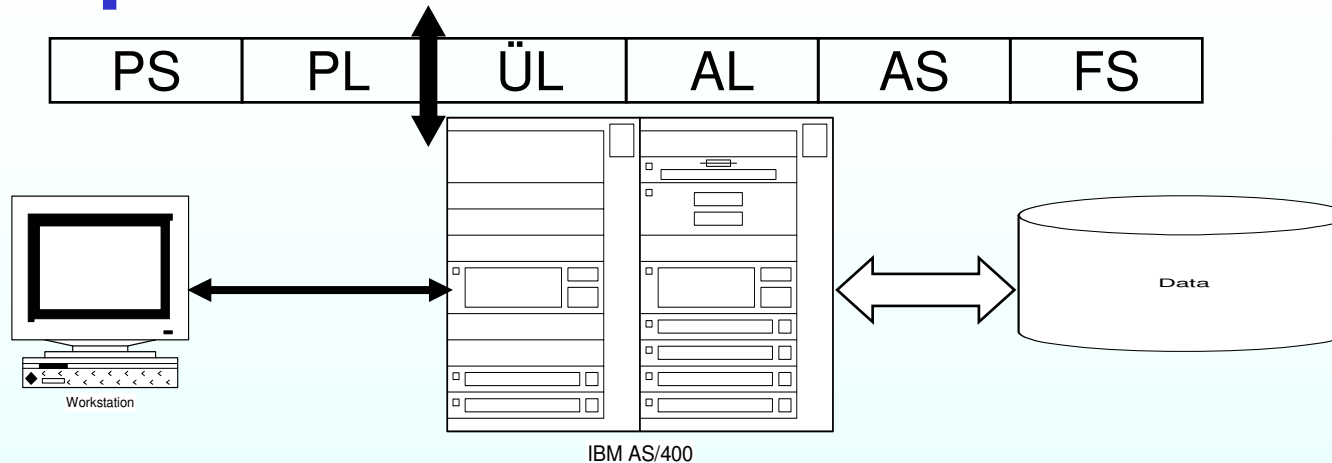
Prezentációs szolgáltatások,  
Prezentációs logika  
Üzleti logika, Adatlogika

## Host

Adatszolgáltatások  
Fájlszolgáltatások

- **Csak az AS és a FS van a szerveren**
  - “Az alkalmazás a kliensen, az adatbázis pedig a szerveren van.”
  - Általában ezt azonosítják magával a C/S architektúrával
- **Hátrányok**
  - Az alkalmazás adminisztrációja bonyolult lehet - ha a kód vagy a konfiguráció megváltozik, azt mindegyik desktopon külön-külön érvényesíteni kell
  - Nagyteljesítményű desktopokra van szükség mert az alkalmazás még mindig a desktopon van - az adatszelekció és - adminisztráció kivételével.

# Távoli prezentáció C/S



## Desktop

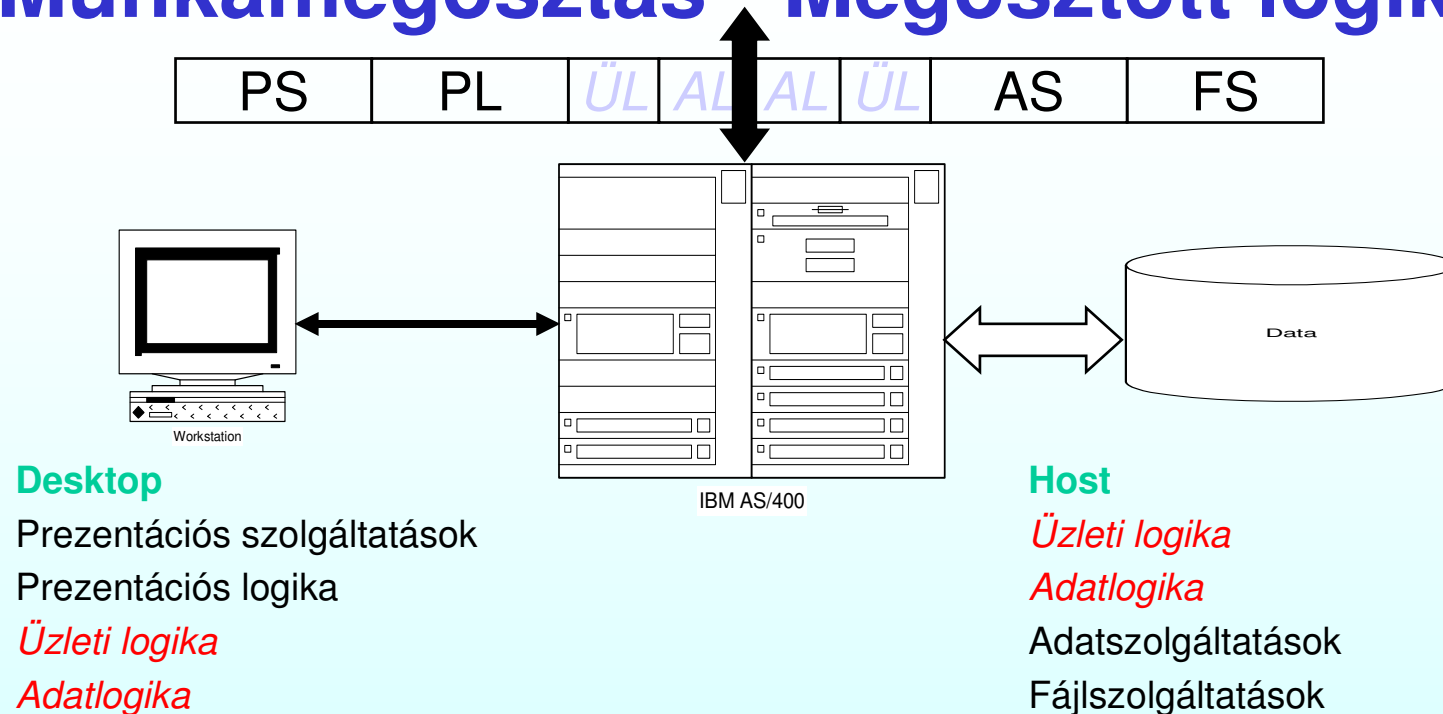
Prezentációs szolgáltatások  
Prezentációs logika

## Host

Üzleti logika, Adatlogika,  
Adatszolgáltatások, Fájlszolgáltatások

- Csak a PS és PL van a desktopon
- A desktopok és a hálózat igénybevétele csökken
- A szerver igénybevétele nő
- Az alkalmazás centralizálttá válik, valamelyest egyszerűsödik az adminisztrációja.
- Ez a konfiguráció - legalább a PS és a PL együtt a desktopon - a minimálisan szükséges feltétele annak, hogy C/S architektúráról beszélhessünk.

# Munkamegosztás - Megosztott logikájú C/S

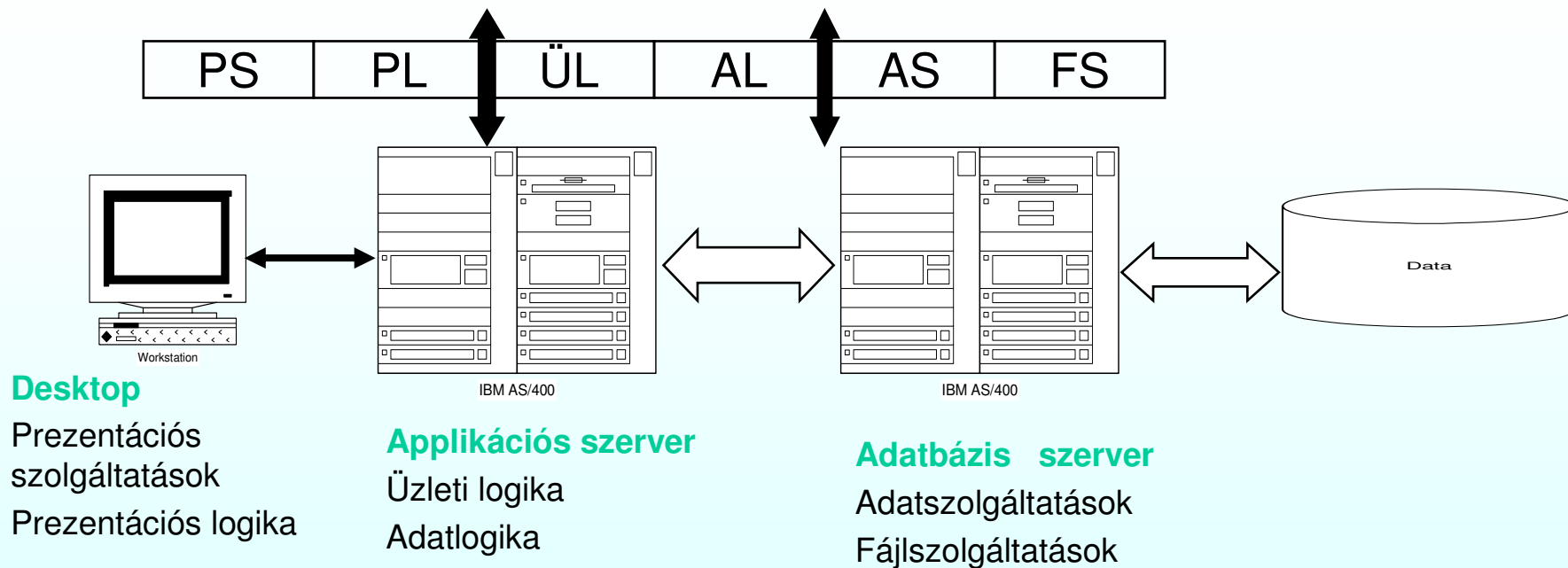


- A desktop és a hálózat terhelése csökkenthető, ha az ÜL és AL egy részét áthelyezzük a szerverre
- A szerver leterheltsége nő
- Az alkalmazás adminisztrációja komplexebbé válik
- Alkalmazás partícionálás: annak eldöntése, hogy az alkalmazás mely részei hol helyezkedjenek el, az alkalmazás tervezőjének a feladata
- A szerveren például triggerek és tárolt eljárások formájában lehet programozási logikát elhelyezni és feldolgozni – távoli eljáráshívás (RCP)

# A kétrétegű C/S architektúrák problémái

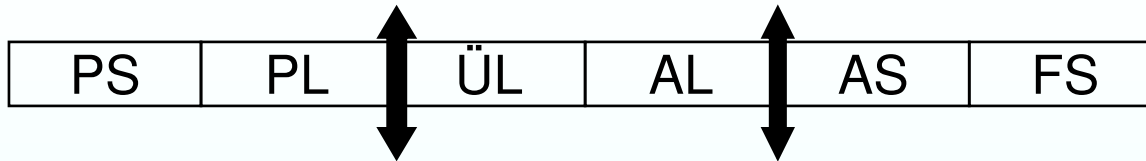
- Sok felhasználó esetén bonyolult az alkalmazás adminisztrációja (pl. minden desktopon át kell vezetni a változtatásokat - telepítés, installálás, tesztelés)
- A tárolt eljárások adminisztrációja szerver oldalon szintén bonyolult lehet - gyakorlatilag megakadályozza, hogy más adatbázis-kezelőre térjünk át, mivel mindegyiknek saját procedurális nyelve van
- Köteget (batch) alkalmazások indítása után a desktop áll annak befejeződéséig
- Sok felhasználó esetén gyakran lassan futnak az alkalmazások

# Háromrétegű C/S



- A háromrétegű C/S megcélazza a kétrétegű C/S architektúrák problémáinak kiküszöbölését
- A kliensen csak a PS és a PL van, plusz egy interfész (API - Application Programming Interface) a középső rétegen futó alkalmazás meghívására
- Az adatbázis serveren csak az AS és a FS fut, így működése a tárolt eljárások menedzselésének kockázata nélkül optimalizálható.
- A középső réteg értelmezi és átalakítja a programtól kapott parancsokat, majd továbbítja az adatbáziskezelő rendszerhez.
- A lekérdezési eredményeket a kliens szintén a szolgáltató rétegen keresztül kapja meg (pl. dinamikus web oldalak)
- (Ha az adatokat ténylegesen adatbázisban tároljuk, hiszen így sok fontos problémára pl. hozzáférési jogok szabályozása, konkurens hozzáférések vezérlésére, hatékony lekérdezés, mentés, kész megoldásokat kaphatunk)

# Háromrétegű C/S



- A középső réteg az alkalmazás szerver, melyen az ÜL és az AS-okat meghívó AL fut.
- Az alkalmazáserver menedzseli a tranzakciókat - heterogén adatbázisok esetén is.
- Az alkalmazás centralizált, így adminisztrációja lényegesen leegyszerűsödik, egyetlen egy helyen kell bármit módosítani.
- A teljes rendszer teljesítménye jobban optimalizálható.
- A feladat-specifikus algoritmusok központi fejlesztése, tárolása: nem kell minden ügyfélben azonos feladatokat megoldani;
- Az alkalmazás szerver eltakarja az ügyfelek elől a felhasznált adatbázisok konkrét típusát, esetleg egyidejűleg több, heterogén technológiájú adatbázist kezelhet, lehetőséget nyújt elosztott tranzakciók megvalósítására;
- Az adatbázis illetéktelen hozzáférések előli elrejtése: az alkalmazás szerver tipikusan a tűzfal környékére magára a tűzfalat megvalósító gépre kerül, így a külvilág közvetlenül csak ezt a kiszolgálót éri el, az adatbázisokhoz csak az alkalmazás szerver férhet hozzá;
- Teljesítménynövekedés, hiszen a feladatot több számítógép között osztottuk fel, cseles programozási technológiákat használva akár dinamikusan allokálhatunk közbülső feladat-specifikus kiszolgálókat a terhelések enyhítésére;
- Egyszerűbb fejlesztés: a megoldás jobban particionált, szabadabban válogathatunk a programozási nyelvek, könyvtárak, protokollok között.

# Háromrétegű C/S - rétegek

PS	PL	ÜL	AL	AS	FS
Prezentációs réteg		Üzleti (alkalmazói) réteg		Adat réteg	

A felhasználói interfész

Az üzleti követelmények (logikai)

Az adatbázis logikai és fizikai megvalósítása

## •A felhasználói interfész

- Tudja, hogyan öntse formába és jelenítse meg a logikai rétegtől jövő adatokat a kliens készülékének megfelelő módon. Az interfész réteg biztosítja a felhasználó hozzáférését a logikai réteg olyan szolgáltatásaihoz is, mint a lekérdezés és az aktualizálás. A felhasználói interfész réteg sosem kommunikál közvetlenül az adatkezelő réteggel, és sohasem végez értéknövelő műveleteket az adatokon. E két funkció mindegyike teljesen a logikai rétegtől függ.

## •Az üzleti követelmények

- Implementálja az alkalmazás ügyviteli szabályait. E réteg nem tartalmaz feltételezéseket, hogyan tárolódik az adat. Az adatrétegtől egyszerűen elvárja, hogy adjon egy általános hozzáférési módot a nyers adatokhoz. A logikai réteg értéket ad a nyers adatokhoz azáltal, hogy alkalmazza rájuk az ügyviteli szabályokat és a harmadik réteg rendelkezésére bocsátja őket.

## •Az adatbázis logikai és fizikai megvalósítása

- Kezeli az adattárolást. Az adatokat a lehető legnyersebb formában kell tárolni. Nincsenek feltételezések, hogyan kell az adatokat feldolgozni, vagy hogy milyen műveleteket kell e rétegben az adatokon végrehajtani.

# Háromrétegű C/S – N rétegű C/S

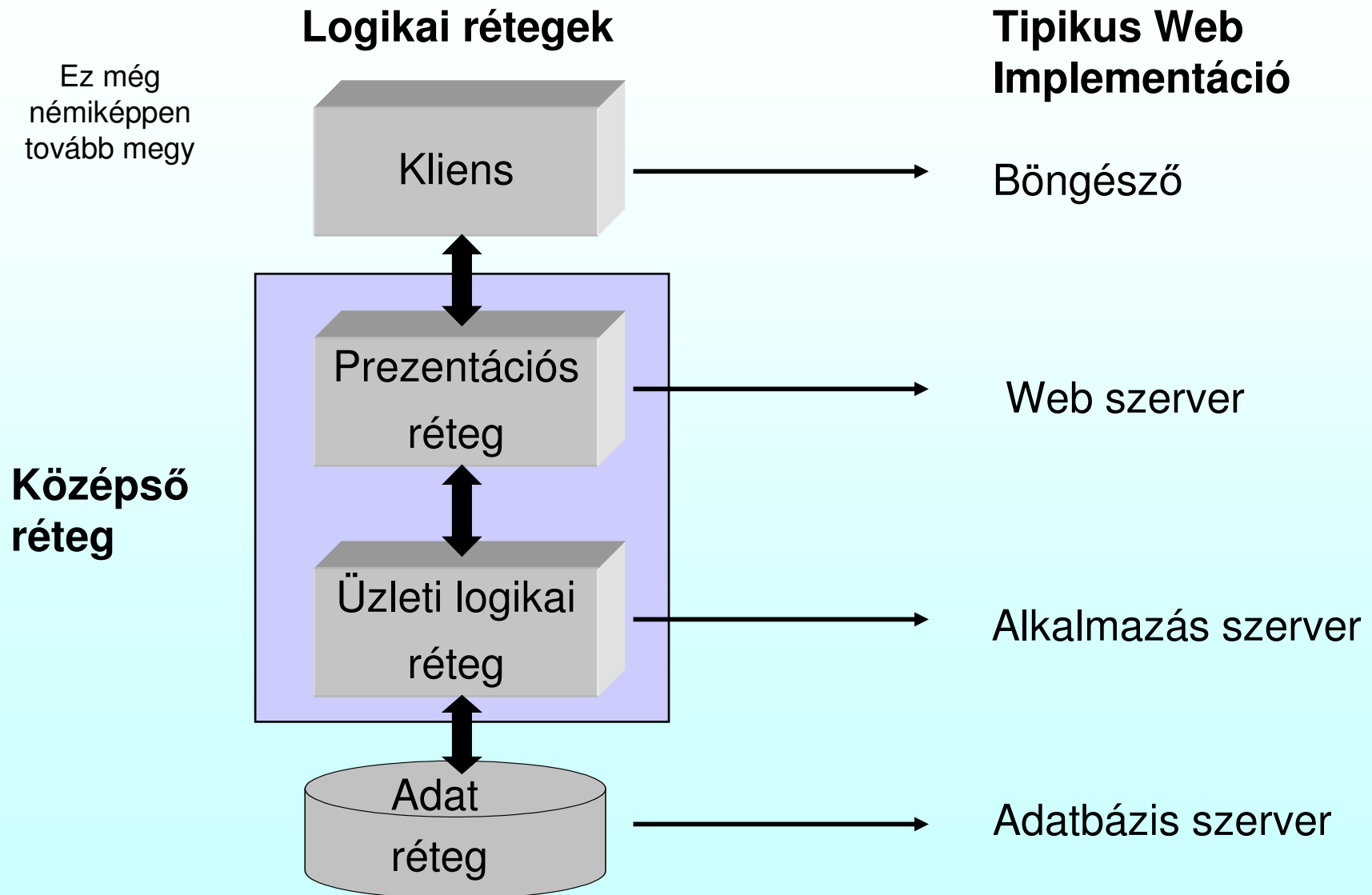
- Az egyes rétegeket alkotó komponensek a feladatukat úgy végzik, hogy azokról nem feltételezik, hogy bármilyen alkalmazás bármilyen részét képeznék. Alkalmazások csak úgy léteznek, mint komponensek együttműködő csoportjai, és mindegyik komponens egyidejűleg sok különböző alkalmazásnak lehet a része
- A háromrétegű architektúra erejét nagymértékben a szemléletváltás adja: komponensek, mint alkalmazások elválaszthatatlan részei, szemben az önállóan létező komponensekkel, melyek alkalmazásoknak szolgáltatásokat tudnak nyújtani. Mivel az alkotórészek önálló szolgáltatóként vannak megtervezve, nem egy alkalmazás beágyazott részeiként, egyszerre lehetnek mindezeknek az alkalmazásoknak a részei.



# Többrétegű C/S

- 5 rétegű C/S
  - Bizonyos felfogás szerint az adat, a logikai és prezentációs rétegek különböző komponenseinek teljes elszigeteléséhez interfész rétegek kellenek az adat- és logikai rétegek, valamint a logikai és prezentációs rétegek között
- N rétegű C/S
  - Más felfogások ok szerint minden egyes réteg sok alrétegre bontható, melyek mindegyike magasabb szintű funkciókat nyújt, mint az alatta fekvő alréteg. Ez a struktúratípus az N-rétegű architektúra.

# Általános Web alapú architektúra



## Kliens réteg

- Gyakorlatilag a browsert jelenti a felhasználó gépén.
- Feladata az adatok megjelenítése és adatbeírási/módosítási lehetőség biztosítása a felhasználó számára.
- A kliens réteg implementálásának módszerei:
  - “Buta” HTML kliens: minden intelligencia a középső rétegben található. Mikor a felhasználó elküld egy weblapot, minden ellenőrzést a középső réteg végez és a hibaüzenetet a felhasználó egy új weblap formájában kapja vissza.
  - Félintelligens HTML/Dinamikus HTML/JavaScript kliens: a weblap tartalmaz némi intelligenciát, ami a kliensen fut le. A kliens végezhet egyszerű ellenőrzéseket (pl. ki vannak-e töltve a kötelező oszlopok).

## A prezentációs réteg

- Weboldalak generál és dinamikus tartalmat tesz az oldalakra.
- A dinamikus tartalom jellemzően adatbázisból származik.
- A réteg másik fontos feladata a kienstől érkező weblapok “dekódolása” - a felhasználó által megadott adatok kiemelése és továbbítása az üzleti logikai rétegnek.

## Az üzleti logikai réteg

- Ebben a rétegben helyezkedik el az alkalmazás logikája (üzleti logika - legalábbis annak legnagyobb része).
- Az üzleti logika végzi:
  - az összes számítást és ellenőrzést,
  - munkafolyamat menedzselést (pl. kapcsolati információk nyomonkövetését),
  - az adathozzáférés vezérlését a prezentációs réteg számára.

## Az adatréteg

- Az adatréteg az adatok menedzseléséért felelős.
- Egyszerű esetben ez lehet egy modern relációs adatbázis. Gyakran azonban biztosítani kell a hozzáférést régi hierarchikus adatbázisokhoz, évtizedek óta futó rendszerekhez, szövegformában kiexportált adatokhoz stb.
- Az adatréteg feladata, hogy biztosítsa az üzleti logikai réteg számára az általa igényelt adatokat, illetve tárolja el az üzleti logikai réteg által küldött adatokat.

# RPC (Remote Procedure Call)

- A hálózaton különböző erőforrások (remote file system, remote nyomtató stb.) elérhetők
- Távoli programok is elérhetők!! A szerver a CPU-ját osztja meg a hálózaton – távoli eljáráshívás (kliens-szervet kapcsolatokban, UDP fölött tipikusan ezt használják)
- Hasonlít a programozási nyelvek eljáráshívásaira és függvényhívásaira: valamilyen paraméterekkel meghívjuk az eljárást és visszakapjuk az eljárás eredményét (a főprogram a konkrét eljárásról semmit nem tud)
- Elképzelt példa: `get_IP_address(host_name)`
  - Elküld egy UDP szegmenst a DNS szervernek
  - Meg várja a választ
  - (szükség esetén újra küldi az UDP szegmenst)

# RPC

- A meghívott eljárás a távoli gépen fut
- A szerver processz nem érzékei, hogy távoli hívás volt
- A progr. nyelvekben vannak távoli eljáráshívásra vonatkozó utasítások, könyvtári függvények (kliens csont – client stub)
- A szerver egy szerver csontkhoz (server stub) kapcsolódik
- Az RCP tipikusan, de nem kizárólagosan az UDP-re épül
- Példa: WWW keresés



# RPC

- Folyamat: (a kérés/válasz ill. paraméterek útja)
  1. A kliensprogram meghívja a kliens stub-ot
  2. Továbbítás a szállítási entitáshoz a kliensen belül
  3. Továbbítás a hálózaton a szerverhez
  4. A szerver szállítási entitása a szerver stub-nak adja át
  5. A szerver processz meghívása
  6. A szerver processz megadja az eredményt a szerver stub-nak
  7. Továbbítás a szállítási entitáshoz a szerveren belül
  8. Továbbítás a hálózaton a klienshez
  9. A kliens szállítási entitása a kliens stub-nak adja át a választ
  10. Az kliensprogram megkapja az eredményt