Az Informatikai Tervező

Tematikus vizsgafelkészítő

Szakmai Útmutató 1.számú Melléklete

Magyar Mérnöki Kamara
Kiadványsorozata FAP-2023/114-HIT

Az Informatikai Tervező

Szakmai Útmutató 1. számú Melléklete

IT Tervező tematikus vizsgafelkészítő

MMK FAP azonosító:
2023/114-HIT

Budapest, 2023. október

A sorozat szerkesztője:

WAGNER ERNŐ
a Magyar Mérnöki Kamara elnöke

Készült a Magyar Mérnöki Kamara Hírközlési és Informatikai Tagozatának gondozásában, a 2023. évi Feladat Alapú Pályázatok pénzügyi keretéből.

Szerző:

Kakuk Ilona

A forrás dokumentumok szerzői, ami alapján ez a dokumentum elkészült:

Dr. Gábori László; Dr. Beinschróth József; Kakuk Ilona; Dr. Molnár Bálint; Nógrádi Gábor; Rátkay Tamás;

Lektorálta:

Nógrádi Gábor

Kiadó:

Magyar Mérnöki Kamara
1117 Budapest, Szerémi út 4.
fap@mmk.hu, www.mmk.hu

TARTALOMJEGYZÉK

Preambulum

Jelen dokumentum a Magyar Mérnöki Kamara (MMK) Informatikai Tervező jogosultság megszerzését támogató szakmai kiadvány sorozat része:

- **IT Tervező tematikus vizsgafelkészítő**: Publikus kiadvány.
- **IT tervező Szakmai Útmutatója kiadvány 1. számú Melléklete**

2023-as sorozat FAP-2023/114-HIT kiadványai, az Informatikai Tervező Szakmai Útmutatója és a vizsgára felkészítő mellékletei a következők:

IT tervező Szakmai Útmutatója; Publikus kiadvány.

Mellékletei

1. **IT Tervező tematikus vizsgafelkészítő**: Publikus kiadvány. (1. számú Melléklet);
2. **IT Tervező Kérdésbank**: Publikus kiadvány. (2. számú Melléklet);
3. **IT Tervező írásbeli Vizsgakérdések-válaszok**: Belső kamarai munkaanyag, nem publikus kiadvány. (3. számú Melléklet);
4. **IT Tervező szóbeli Vizsgakérdések-válaszok**: Belső kamarai munkaanyag, nem publikus kiadvány. (4. számú Melléklet);

1 Bevezető

Ez az „IT Tervező Tematikus vizsgafelkészítő” dokumentum az „IT Tervező Szakmai Útmutató” 1. számú melléklete, ami a Magyar Mérnöki Kamara által kibocsátható -a 618/2021. (XI. 8.) Korm. rendelet és a 266/2013. (VII. 11.) Korm. rendeletek szerint- a létfontosságú rendszerek és létesítmények informatikai beruházásaihoz szükséges informatika tervek készítését és ellenőrzését végző „Informatikai Tervező” szakmagyakorlási jogosultsági vizsga megszerzéséhez készült felkészítő tudástár.

A felkészítő tudástár tartalmazza a vizsga valamennyi kérdéskörét, jellemzően rövid, célzottan 1 oldal / témakör alapú, és könnyen feldolgozható kivonat alakjában. A korszerű anyag a vizsgán felmerülő kérdésekhez teljes körű támogatást biztosít.

A vizsga is az itt bemutatott tematika és kategóriák szerint épül fel, így **azok, akik legalább egyszer, figyelmesen áttanulmányozzák a tematika szerint a Kérdésbankot és ezt a felkészítő anyagot, sikerre számíthatnak**.

2 Tematika-Kategóriák

A Tematika a tervezés koncepcionális elemeire helyezi a hangsúlyt, a

Fogalmak – Megközelítések – Menedzsment

ismereti területek mentén, és mindegyikben a gyakorlati tudnivalókat előtérbe helyezve hoz létre **átfogó tervezői/tervezési orientációt**.

A Tematika felépítése a következő megfeleltetéseket mutatja az alapkoncepcióval :

Koncepció	Tematika
Fogalom	**Szakmai fogalom tár**
Megközelítés	**Tervezői tudásbázis**
Menedzsment	**Tervezési környezet**

A magas szintű Tematika a **Kategóriák**, **Tárgykörök** mentén az Enterprise Architecture, valamint a Software Architecture fogalomkörében mozogva alakít ki specifikus ismeret-csoportokat. A sort a modellezés és az irányítás (Governance) vonatkozásai egészítik ki, egészen a törvényi előírások bemutatásáig.

Tematika	Kategória	Tárgykör
Szakmai fogalom tár	**Fogalom**	• Módszertan • Technológia • Implementáció • Architektúra • Biztonság • Szabályozási környezet • Menedzselés • Tervező feladata

Tematika	Kategória	Tárgykör
Tervezői tudásbázis	• **Tervezési módszertan** • **Technológiai eszköztár** • **Tervezési minta készlet** • **Koncepció**	• Vállalati Architektúra • Szoftver Architektúra • Harwdare Architektúra • Megoldás Architektúra • Alkalmazás Architektúra • Architektúra keretrendszer • Rendszertervezés • Szoftverfejlesztés • Tervezés • Adatkezelési technológiák • Integrációs technológiák • Alkalmazási rendszerek • Hálózati technológiák • Biztonsági technológiák • Hardware technológiák • Alapelvek • Szemlélet
Tervezési környezet	• **Modellezés** • **Irányítás**	• Tervezési keretrendszer • Fejlesztői platform • Projektvezetési módszertan • Fejlesztési módszertan • Kommunikációs elvek

A Tematika szerinti kategória besorolás megegyezik a felkészítő anyag, az írásbeli, és a szóbeli vizsgán feltehető kérdések besorolásával.

Az írásbeli kérdéseket a kamarai vizsgáztató rendszer témakörök alapján automatikusan választja ki a vizsga során, az itt megadott témakörök szerinti kérdések közül.

3 Architektúra kialakítás alapelvei

Az IT architektúra kialakításának alapelvei az IT szervezetek által alkalmazott tervezési elvek, amelyek segítenek az informatikai rendszerek hatékonyabb és biztonságosabb működtetésében. Az alábbiakban felsorolok néhány alapelvet:

Standardizáció: Az IT architektúra kialakításának egyik alapelve a standardizáció, amely azt jelenti, hogy az IT rendszerek és alkalmazások kialakítása során az iparágban elfogadott és bevált szabványokat és protokollokat kell követni. A standardizáció csökkenti a rendszerek és alkalmazások összeomlásának és hibáinak valószínűségét, és lehetővé teszi a különböző rendszerek és alkalmazások hatékonyabb integrálását.

Modularitás: Az IT rendszerek és alkalmazások moduláris kialakítása azt jelenti, hogy azokat kisebb, önállóan működő egységekre kell bontani, amelyek könnyen integrálhatók és karbantarthatók. A moduláris kialakítás lehetővé teszi a rendszerek és alkalmazások gyorsabb fejlesztését, tesztelését és bevezetését.

Megbízhatóság: Az IT rendszerek és alkalmazások megbízhatósága azt jelenti, hogy képesek a folyamatos működésre, és minimálisra csökkentik az összeomlások, a szünetek és az adatvesztések kockázatát. A megbízhatóság növeli az IT rendszerek és alkalmazások felhasználói elégedettségét és bizalmát.

Rugalmasság: Az IT architektúra kialakítása során fontos szempont a rugalmasság, vagyis az alkalmazkodó képesség az új technológiákra, az üzleti követelmények változásaira és a felhasználói igényekre. Az IT architektúra rugalmassága lehetővé teszi az alkalmazások és rendszerek hatékonyabb és gyorsabb alkalmazkodását az üzleti változásokhoz.

Egyszerűség: Az IT architektúra kialakítása során az egyszerűség elve fontos, hogy a rendszerek és alkalmazások könnyen érthetők, karbantarthatók és fejleszthetők legyenek. Az egyszerűség elve csökkenti a redundancia és az ismétlődés lehetőségét, valamint javítja a rendszerek és alkalmazások hatékonyságát.

Biztonság: Az IT architektúra kialakításánál kiemelkedő fontosságú az adatok és rendszerek biztonsága. Az IT architektúra kialakítása során fontos figyelembe venni a biztonsági előírásokat és elveket, és azokat az alkalmazások és rendszerek kialakításába beépíteni.

Skálázhatóság: Az IT architektúra kialakításánál fontos szempont a skálázhatóság, vagyis az alkalmazások és rendszerek könnyű bővíthetősége és javíthatósága. Az IT architektúra kialakítása során figyelembe kell venni a jelenlegi és jövőbeli igényeket, és olyan architektúrát kell kialakítani, amely könnyen bővíthető és skálázható.

Interoperabilitás: Az IT architektúra kialakítása során fontos figyelembe venni az interoperabilitás elvét, vagyis az alkalmazások és rendszerek képességét arra, hogy más rendszerekkel és alkalmazásokkal kommunikáljanak. Az interoperabilitás elve lehetővé teszi az adatok és az információk hatékonyabb megosztását, valamint az egymástól független rendszerek és alkalmazások együttműködését.

3.1 Architektúra központú szemlélet

Az Architektúra Központú Szemlélet az alkalmazások és rendszerek tervezésének és fejlesztésének egyfajta megközelítése, amelyben az architektúra a középpontban áll. Az architekturális szemlélet azt jelenti, hogy az alkalmazások és rendszerek tervezése során az architekturális szempontokat kell előtérbe helyezni, és az architektúra tervezését a folyamat elején kell elkezdeni.

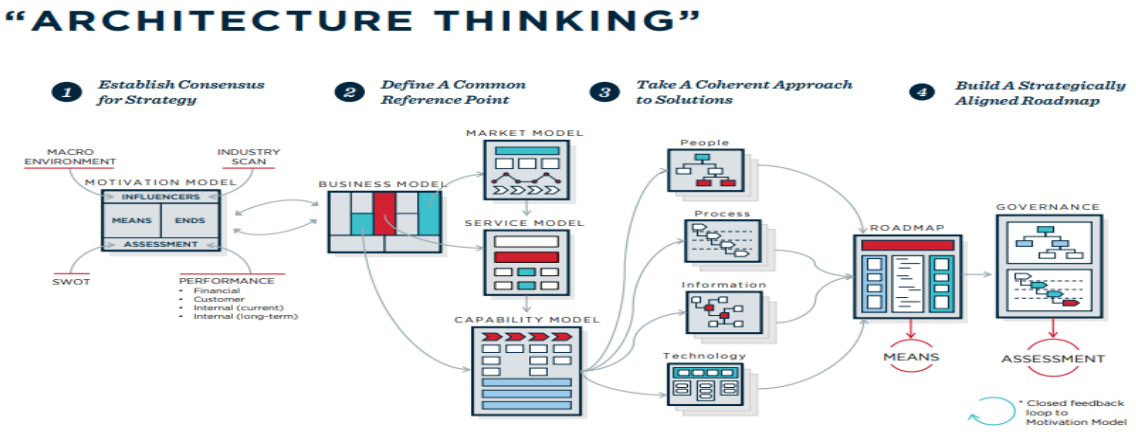
Az Architektúra Központú Szemlélet elemei

Alaposan kidolgozott architektúra: Az architekturális szemléletben az architektúrát előre meg kell tervezni és kidolgozni. Az architektúra tervezésénél figyelembe kell venni az alkalmazások és rendszerek funkcionális és nem funkcionális jellemzőit, és olyan architektúrát kell kialakítani, amely lehetővé teszi az alkalmazások és rendszerek hatékony működését.

Az architektúra tervezése előtt történő tervezés: Az architekturális szemléletben az architektúrát a tervezés folyamatának elején kell tervezni. Az architekturális szemlélet azt jelenti, hogy az architektúrát nem az alkalmazások és rendszerek tervezése után kell tervezni, hanem az alkalmazások és rendszerek tervezése előtt.

Az architektúra folyamatos javítása: Az architekturális szemléletben az architektúrát folyamatosan kell javítani és fejleszteni. Az architektúra tervezésekor figyelembe kell venni a jelenlegi és jövőbeli igényeket, és olyan architektúrát kell kialakítani, amely lehetővé teszi az alkalmazások és rendszerek folyamatos fejlesztését és javítását.

Az architekturális szemlélet lehetővé teszi az alkalmazások és rendszerek hatékonyabb és megbízhatóbb tervezését és fejlesztését.



3.2 Az „IT - Informatikai Tervező” humán tényezői

Habár az "informatika" rendkívül széles spektrum, és az információ digitális átalakításától, a gépi automatizáláson át az interaktív alkalmazásokon keresztül a mesterséges intelligencia speciális területéig sok mindent felölel, ám bármelyik szeletét nézzük is, mindig az informatikát művelő ember a legfontosabb tényezője.

Az IT Tervező szerepkör fő jellemzője a különböző szintű megoldási tervek, és megoldási tevékenységek meghatározása, kidolgozása, folyamatos ellenőrzése, illetve ezen tevékenységek projekt szintű irányítása.

Informatikai Tervező

Egy Informatikai Tervező (Enterprise Architect vagy ICT architect) magas szintű informatikai megoldásokat hoz létre üzleti alkalmazásokhoz, portfóliókhoz, vállalati vagy közszolgáltatási rendszerekhez stb. Ezek a megoldások segítenek az IT-cégeknek a kommunikáció, a biztonság, a hálózatépítés, a tárolás stb. tervezésében és kezelésében. Az informatikai tervezőnek egyaránt rendelkeznie kell megfelelő magasszintű elméleti képzettséggel, és gyakorlati ismeretekkel.

3.2.1 Szakmai szempontrendszer

A tervezés a szakmai ismeretek átgondolt alkalmazásával egy megoldási elképzelés kidolgozása egy összetett problémára.

A digitális világ kihívásainak összetettsége a magas szintű fogalmi gondolkodás és szakmai tapasztalat egymást átható ötvözetének talaján maga is komplex válaszokat igényel, ami megoldási elemek és összefüggések sokaságát generálja, mindezek megtervezőjétől pedig speciális **szakmai (Hard Skill)** és **személyiség béli (Soft Skill)** képességeket, tulajdonságokat kíván.

Szakmai szinten az Informatikai Tervező legfontosabb eszköze a mindennapi bonyolultság uralásához egy összetett szemléletmód elsajátítása és folyamatos építése, amely néhány jelentős összefüggés, módszer, szabály ismerete alapján egyaránt képessé teszi a probléma megértésére, hatékony elemzésére, valamint a megoldás megalkotására, fejlesztésére, létrejöttének biztos nyomon követésére és megvalósítására.

A személyiség szintjén a szakmai megalapozottságú tervezés realizálásához szükség van a terv széles körű elfogadtatására is azok részéről, akik megvalósításában résztvevőként érdekeltek. A **tervezői attitűd** így a szakmai tudás és a személyiség jegyek egyfajta ötvözeteként írható le.

3.2.2 Készségek

Mivel a tervezés alapvetően szellemi, gondolati alapú konstruáló tevékenység, ezért elsősorban speciális értelmi képességekre és megfelelő képzettségre, tapasztalatra épül.

Az alábbi jellemzők az Informatikai Tervező szerepkört ellátó jelölt elvárt személyiség jellemzőinek, szemléletének meghatározásai, az ún. „Soft Skill”-ek.

3.2.2.1 Kategorizáló, osztályozó képesség

Az összetett problémák alapvető kezelési módjának, egy probléma kezelhető részekre bontásának ismerete és gyakorlata. A felbontás módszere a jól megkülönböztethető jellemzők szerinti csoportosítás.

A logikai csoportosítás módjai

Képesség	Jellemző
Kategorizálás	Csoportképzés (diszjunkt szempontú csoportosítás)
Osztályozás	Viszony képzés

3.2.2.2 Elemzési képesség

Egy probléma megértéséhez szükséges általános megismerési tevékenység ismerete. Az elemzés a jelenség szintjén, a teljességből bontja ki a mélyebb összefüggéseket, az elemi részleteket, majd ezekből modellek útján építi fel ismét az így megismert egész lényegét.

Az elemzés módjai

Képesség	Jellemző
Elvonatkoztatás	Lényeglátás, absztrahálás, fogalmi gondolkodás
Dekomponálás - Komponálás	Viszonyrendszer kialakítás és nézőpont váltogatás.
Modellezés	Szemléltetés, megjelenítés, formalizálás

3.2.2.3 Alkotási, tervezési képesség

A sok szempontból kiegyensúlyozott, optimális megoldás tartós, következetes, tervszerűen kialakított gondolkodási tevékenység eredménye. A szisztematikus alkotói építkezés sok előrelátást, hatékony képzelőerőt, szilárd szemléletet és közvetlen személyiséget követel.

Alkotási jellemzők

Képesség	Jellemző
Koncepcionalitás	Rendszerszemlélet, látásmód
Innovativitás	Tájékozottság, rugalmasság, egyediség
Objektivitás	Józanság, viszonyítási képesség

3.2.2.4 Értékelési, minősítési képesség

A tervezés műszaki jellegű tevékenység, ami fokozottan érvényes az Informatikai tervezés esetében, így minden részletében meg kell jelennie az eredmény jellemzők változatos, mérés alapú értékelésének.

Értékelési jellemzők

Képesség	Jellemző
Precizitás	Gondosság, pontosság
Kvalitatív attitűd	Mérnöki gondolkodás, tendenciózusság

3.2.2.5 Ismeretátadási, kommunikációs képesség

A tervek megfelelő hatékonyságú bemutatása biztosíthatja az elfogadásukat, aminek hiánya viszont a valószínűsíthető kudarc előjele.

Kommunikációs jellemzők

Képesség	Jellemző
Verbalitás	Ismeret átadási képesség, megjelenés, didaktikusság
Szuggesztivitás	Közvetlenség, határozottság, fellépés, impulzivitás

3.2.2.6 Szervezői, irányítási képesség

A jóváhagyott terv megvalósítása csak gondosan tervezett és szervezet munkafolyamatok együttműködésével eredményes. A munkaszervezés, irányítás érzékeny feladat a humán erőforrás sokrétű jelenléte, és szerteágazó kompetencia szintje miatt.

Vezetői jellemzők

Képesség	Jellemző
Együttműködés	Kompromisszum készség, érdekérvényesítő képesség
Vezetés	Döntésképesség, mérlegelési képesség, körültekintés

4 Modellezés

A modell egy absztrakt reprezentációja a valós világnak vagy egy részének, amelyet a modellező egy adott cél érdekében hoz létre. A modell lehetővé teszi az adott rendszer vagy folyamat szimulációját, elemzését vagy tervezését azáltal, hogy leegyszerűsíti vagy absztrahálja a valós világ összetett jelenségeit.

A modellek lehetnek fizikai vagy absztrakt jellegűek, és különböző területeken alkalmazzák őket, mint például a fizika, az informatika, az üzleti tervezés, a szoftvertervezés, a biológia és az urbanisztika. A modell különböző szintjei lehetnek, az absztrakttól a részletes szintig, és eltérő célokra használják őket.

A modellezési technikák lehetővé teszik a modell készítését és elemzését. A modellek lehetnek matematikai vagy grafikus alapúak, és a modellező különböző eszközöket és technikákat használhat, mint például a matematikai egyenleteket, a szimulációkat, a rendszerszintű diagramokat és az UML-t (Unified Modeling Language).

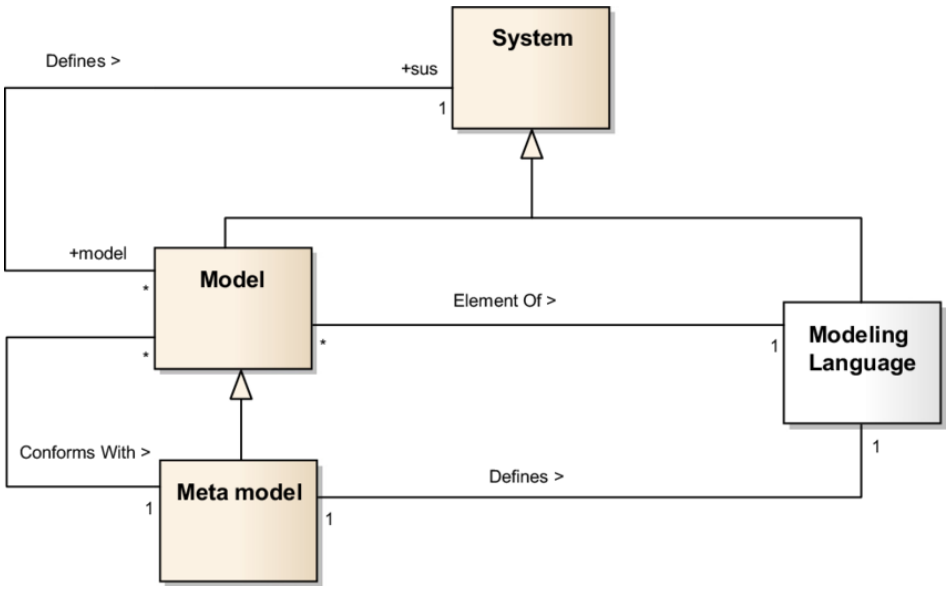
A modell fontos szerepet játszik a problémák megoldásában és a tervezési folyamatban, mivel lehetővé teszi az adott rendszer vagy folyamat előzetes elemzését és tervezését, mielőtt az ténylegesen megvalósulna. A modellek alkalmazása javítja a hatékonyságot, növeli a pontosságot és csökkenti a tervezési hibák kockázatát.

4.1.1 Metamodell

A metamodell egy olyan modellezési eszköz, amely lehetővé teszi egy adott modell absztrakt szintjének leírását és definiálását. A metamodell maga is egy modell, amely leírja egy adott modell alapvető elemeit, relációit és szabályait.

A metamodell segítségével lehetőség van az adott modell által használt elemek és szabályok egyértelmű definiálására és formális leírására. Ezáltal a metamodell lehetővé teszi a modell konzisztens és szabványos leírását, amely segíti az eltérő modellek közötti összehasonlítást, azok összefűzését és a modell általánosabb értelmezését.

A metamodell alkalmazása széles körben elterjedt az informatikában, a szoftvertervezésben, az adatmodellezésben és más modellezési diszciplínákban. A metamodell alapú megközelítés lehetővé teszi az eltérő modellek és modellezési nyelvek közötti integrációt és szabványosítást, amely javítja a modell alapú rendszerek hatékonyságát, minőségét és karbantarthatóságát.

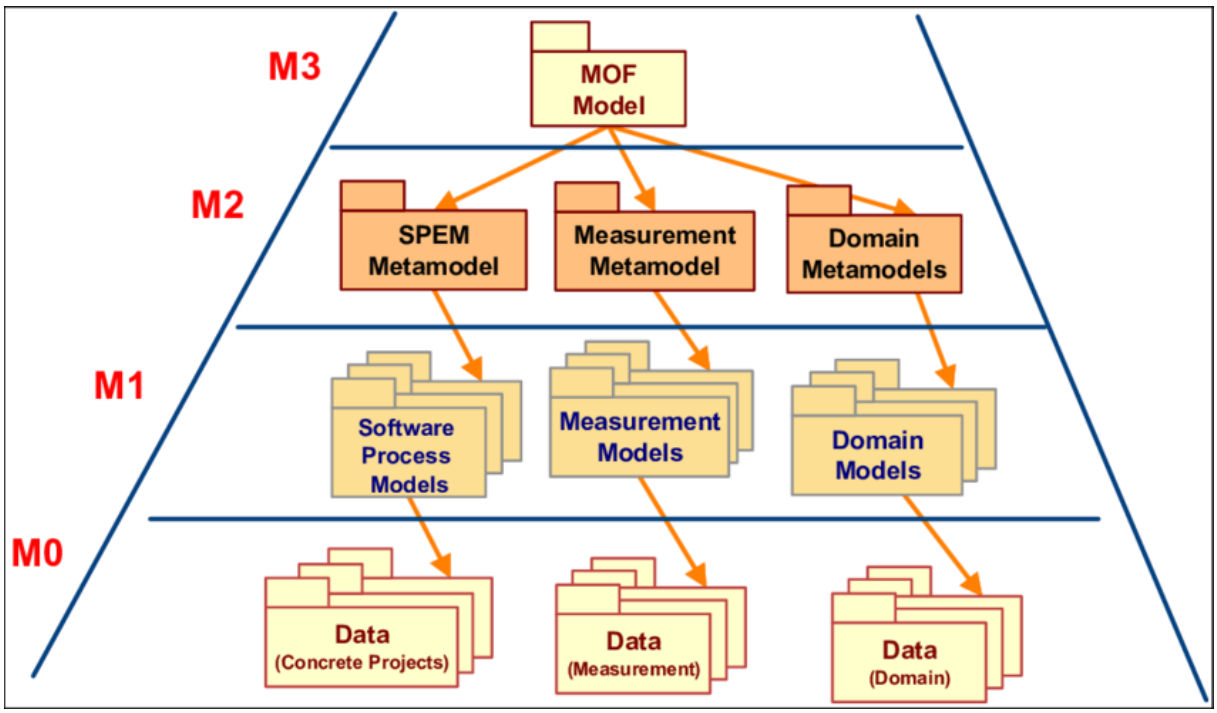


4.1.2 Meta-metamodell

A meta-metamodell egy olyan modellezési eszköz, amely maga a metamodell leírása és definiálása. A meta-metamodell a metamodell maga, és lehetővé teszi azon modellek definiálását és formális leírását, amelyek a metamodell alapján készülnek.

A meta-metamodell egy absztrakt szintű modellezési eszköz, amely lehetővé teszi az adott modellek és metamodell-ek közötti hierarchikus felépítést és kapcsolatokat. Ezáltal a meta-metamodell lehetővé teszi az összetettebb modellek és modell elemek definiálását, amelyek szabványosított és formális leírását használják.

A meta-metamodell általánosan alkalmazható az informatikában, a szoftvertervezésben és más modellezési területeken, amelyek széles körű és hierarchikus modellek alkalmazását igénylik. A meta-metamodell lehetővé teszi a modellezési nyelvek és modellezési eszközök közötti integrációt és szabványosítást, amely javítja a modellek hatékonyságát, minőségét és karbantarthatóságát.



4.2 Tervezési jelölésrendszerek

Az IT tervezési jelölések vagy jelölésrendszerek olyan eszközök, amelyek segítik az alkalmazások és rendszerek tervezését, dokumentálását és kommunikációját.

Jelölés rendszerek

UML (Unified Modeling Language): Az UML az egyik legelterjedtebb jelölés rendszerek a szoftvertervezésben. Az UML segít az alkalmazások és rendszerek különböző szempontjainak, mint például az adatok, a funkciók és az architektúra, leírásában és megjelenítésében.

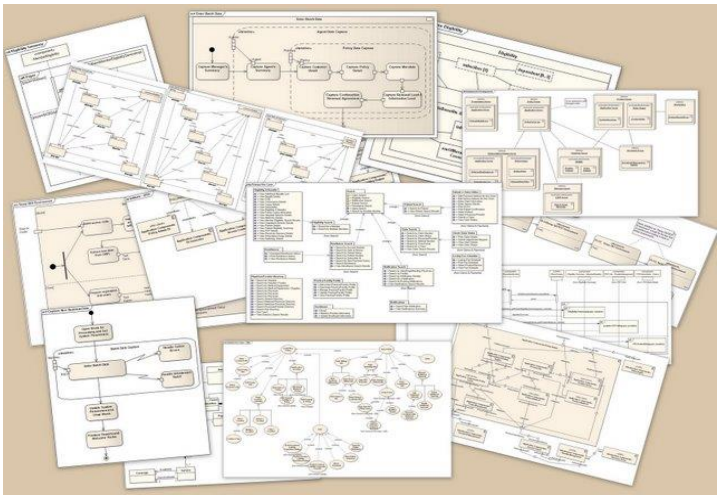
ERD (Entity-Relationship Diagram): Az ERD segít az adatbázis-tervezésben, és ábrázolja az adatbázisban szereplő entitások, attribútumok és kapcsolatok közötti kapcsolatokat.

DFD (Data Flow Diagram): A DFD segít a folyamatok és adatok áramlásának ábrázolásában egy rendszerben. A DFD ábrázolja a folyamatokat, az adatok áramlását és a rendszerrel való kapcsolatokat.

BPMN (Business Process Model and Notation): A BPMN segít a vállalati folyamatok ábrázolásában. A BPMN segítségével a vállalati folyamatokat ábrázolhatjuk, kezelhetjük és optimalizálhatjuk.

Architektúra diagramok: Az architektúra diagramok segítenek az alkalmazások és rendszerek architektúrájának ábrázolásában. Az architektúra diagramok általában a komponenseket, az interfészeket és a kapcsolatokat ábrázolják.

Az IT tervezési jelölések és jelölés rendszerek használata javítja az alkalmazások és rendszerek hatékonyságát és minőségét. Az IT tervezési jelölések és jelölés rendszerek megkönnyítik a tervezési folyamatot, az áttekinthetőséget és a kommunikációt a projektben résztvevők között.



4.3 Tervezési keretrendszer

4.3.1 Szoftvertervezési keretrendszer

A szoftvertervezési keretrendszerek olyan előre meghatározott struktúrák és irányelvek, amelyek segítenek a szoftvertervezés folyamatának hatékonyabbá tételében. Ezek a keretrendszerek általában előre elkészített tervezési sablonokat, modell- és architekturális mintákat, illetve szoftvertervezési szabályokat tartalmaznak, amelyek segítségével a fejlesztők egyszerűbben és hatékonyabban hozhatnak létre jó minőségű szoftvereket.

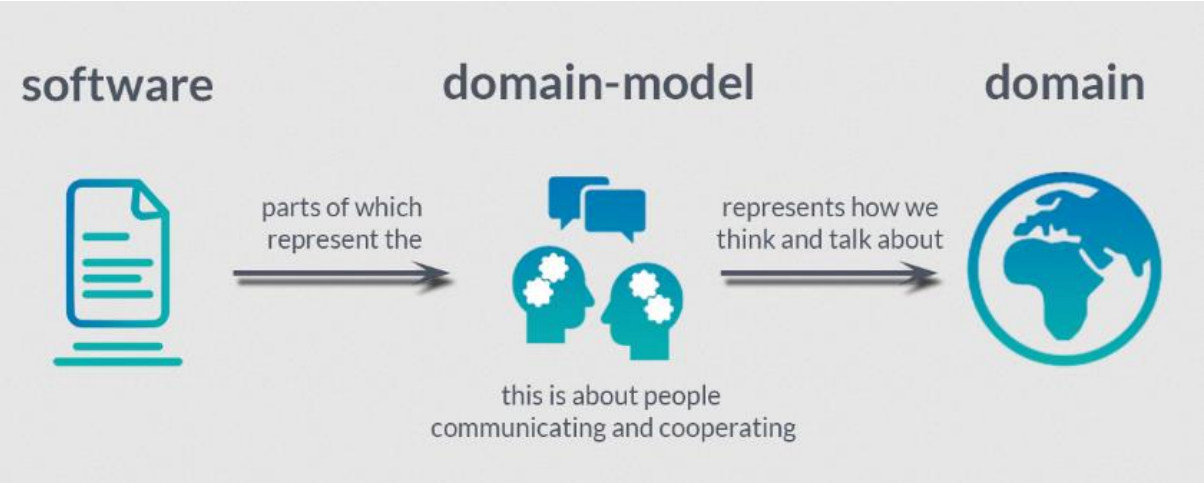
A szoftvertervezési keretrendszerek különböző típusai léteznek, amelyek az egyes szoftverfejlesztési szakaszokra fókuszálnak. Néhány példa a szoftvertervezési keretrendszerekre:

Agile: Az Agile keretrendszer a rugalmas és folyamatos szoftvertervezési módszerekre összpontosít. A fejlesztők folyamatosan kommunikálnak és együttműködnek a csapat tagjaival, hogy hatékonyan dolgozzanak az új szoftverfunkciók fejlesztésén.

Waterfall: A Waterfall keretrendszer lineáris szoftvertervezési módszert alkalmaz, amelyben a fejlesztők a következő szakaszra csak az előző szakasz teljes befejezése után léphetnek. Ez a módszer elősegíti az erős dokumentáció elkészítését és a szervezettséget, de korlátozhatja az agilitást.

Model-View-Controller (MVC): Az MVC keretrendszer a modell, a nézet és a vezérlő egységek felhasználásával a szoftveralkalmazás architektúráját határozza meg. Ez a módszer egyszerűsítheti a fejlesztést és a karbantartást, különösen a webes alkalmazások esetén.

Domain-driven design (DDD): A DDD keretrendszer a szoftvertervezést az üzleti szempontok figyelembevételével közelíti meg, és az üzleti logika központi helyzetbe kerül. Ez a módszer lehetővé teszi a fejlesztők számára, hogy a szoftver alkalmazásait az ügyfelek igényeire szabják.



4.3.2 Modellezési keretrendszer

A modellezési keretrendszer (Modeling Framework) egy olyan szoftvertervezési eszköz, amely előre meghatározott sablonokat, struktúrákat, irányelveket és modellezési nyelveket tartalmaz. Segítségével a fejlesztők egyszerűbben és hatékonyabban hozhatnak létre modelleket, amelyek segítségével a szoftverrendszert megtervezhetik és dokumentálhatják.

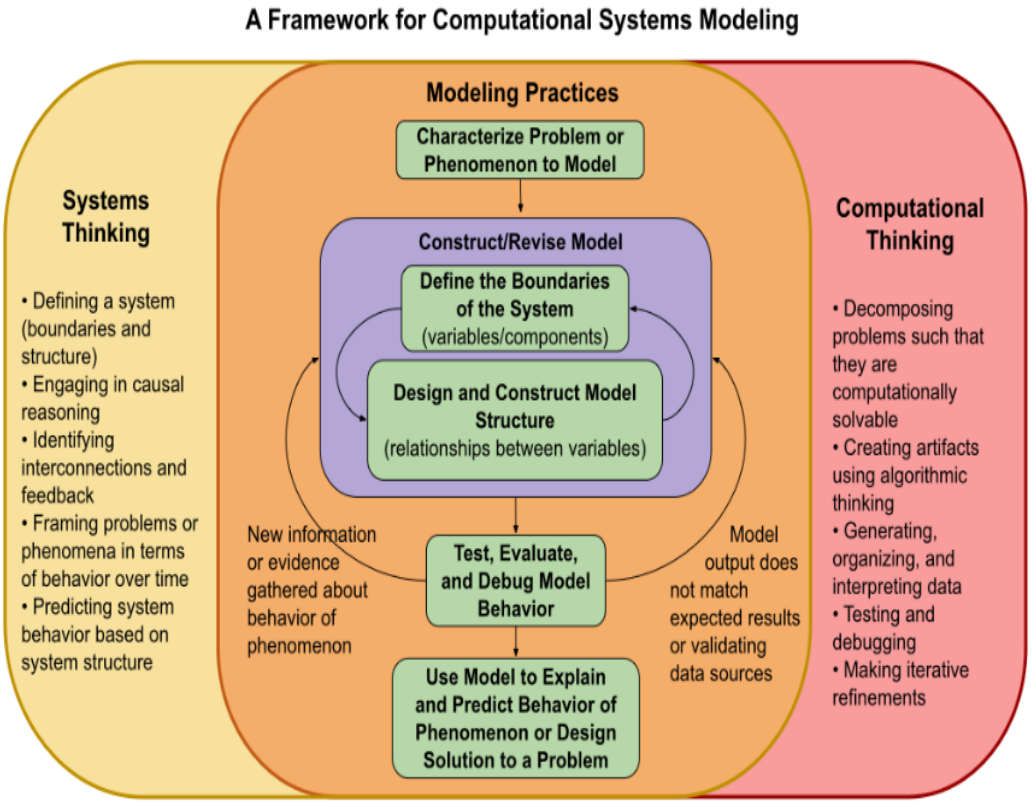
A modellezési keretrendszerek általában bizonyos célokra lettek létrehozva, például objektumorientált szoftvertervezésre vagy folyamat vezérelt modellezésre. Ezek a keretrendszerek előre elkészített sablonokat és modellezési nyelveket tartalmaznak, amelyek lehetővé teszik a fejlesztők számára, hogy gyorsan és hatékonyan hozzanak létre modelleket a szoftverrendszer különböző részeire.

A modellezési keretrendszerek általában lehetővé teszik a modellek automatikus kódgenerálását is, amely nagyban leegyszerűsíti a szoftverfejlesztési folyamatot. A keretrendszerek általában támogatják az összetett modellek elkészítését, beleértve az összetett viselkedési és interakciós modelleket is.

A modellezési keretrendszerek széles körben használatosak a szoftvertervezési iparban, és számos előnyt kínálnak, mint például:

- Gyorsabb és hatékonyabb szoftvertervezési folyamatok
- Jobb dokumentáció és átláthatóság a szoftverrendszerről
- Az erőforrások jobb kihasználása a fejlesztési folyamat során
- Nagyobb skálázhatóság és rugalmasság a szoftverfejlesztési projektekben.

A modellezési keretrendszerek nagyon fontos eszközök a szoftvertervezési folyamatok során, és hatékonyabbá és eredményesebbé teszik a szoftverfejlesztést.



4.4 Szoftvertervezési eszköz

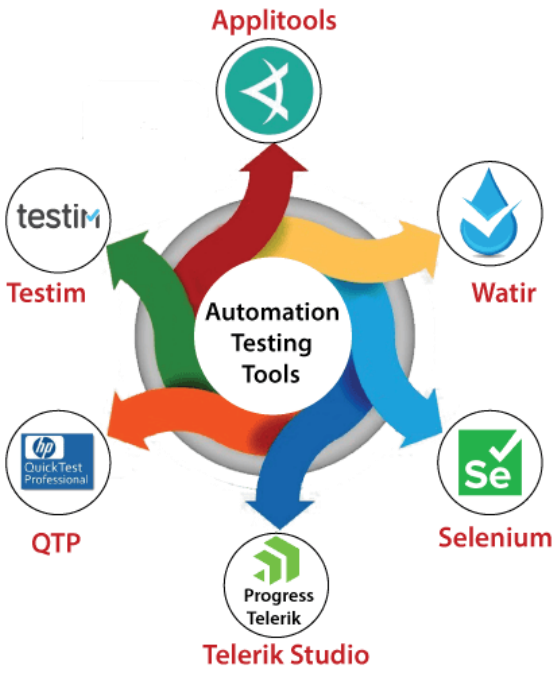
A szoftvertervezés során számos eszköz áll rendelkezésre, amelyek segítik a tervezők munkáját. Néhány ilyen eszköz a következő:

Diagramszerkesztő eszközök: A diagramok fontosak a szoftvertervezés során, mivel segítségükkel könnyen áttekinthetővé válnak az architektúrák, a folyamatok és a kapcsolatok. Ilyen eszköz például a Microsoft Visio vagy az Enterprise Architect.

Kódgeneráló eszközök: Ezek az eszközök a tervezési modell alapján képesek kódot generálni, ami nagyban megkönnyíti a fejlesztők munkáját. Ilyen eszközök a Rational Rose vagy a Visual Paradigm.

Verziókezelő rendszerek: Ezek az eszközök a szoftverfejlesztés során segítenek a különböző verziók nyomon követésében, a változások dokumentálásában és a csapatmunka koordinálásában. Ilyen eszközök például a Git, a Subversion vagy a Team Foundation Server.

Tesztautomatizálási eszközök: Ezek az eszközök automatizálják a tesztelési folyamatokat, ami időt és pénzt takarít meg, és javítja a szoftver minőségét. Ilyen eszközök például a Selenium, a HP QuickTest Pro vagy a TestComplete.



Kódellenőrző eszközök: Ezek az eszközök segítenek az előre meghatározott kódminőségi szabályok betartásában, és az esetleges hibák és biztonsági résfalak felderítésében. Ilyen eszközök a SonarQube, a Checkstyle vagy a PMD.

Projektmenedzsment eszközök: Ezek az eszközök a szoftvertervezés és a fejlesztés teljes életciklusát lefedik, segítik a projektmenedzsmentet, az erőforrások nyomon követését, az időbeosztást és a dokumentációt. Ilyen eszközök például a JIRA, az Asana vagy a Trello.

4.5 Adatmodellező eszköz

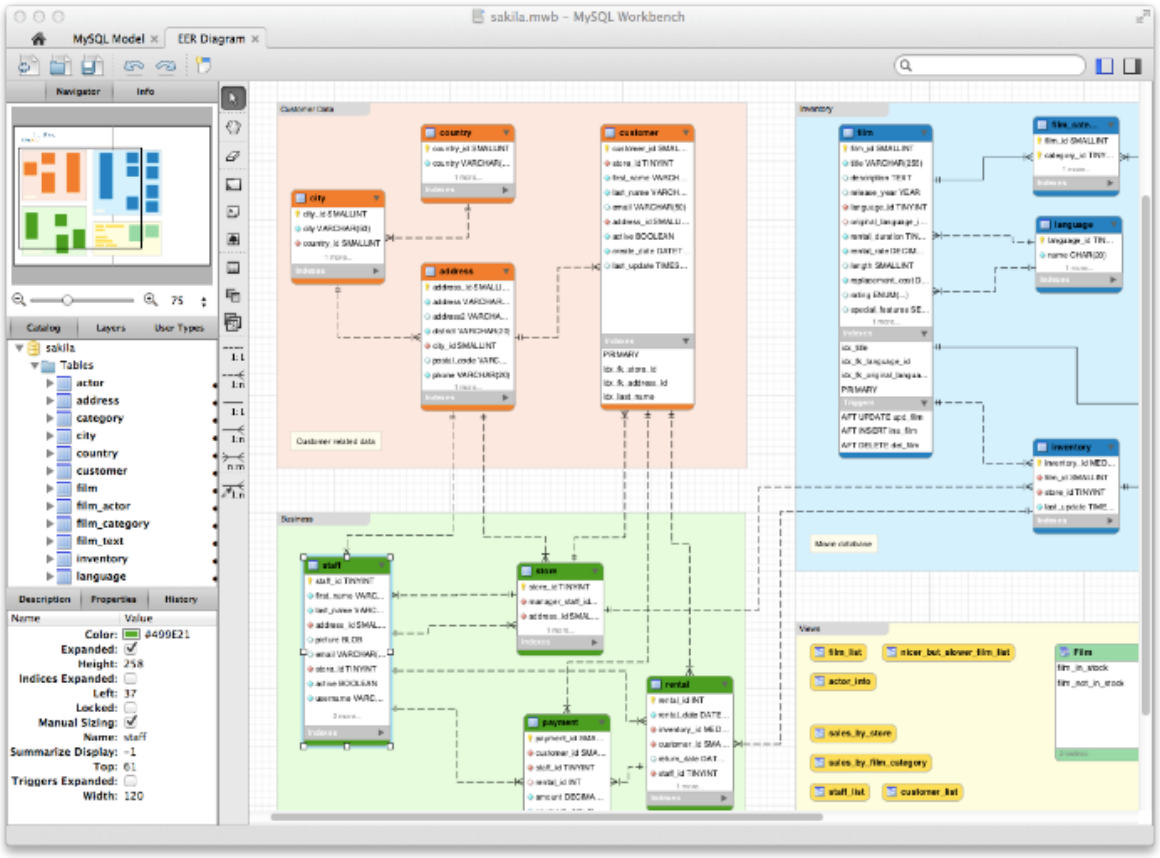
Az adatmodellezés egy fontos szerepet játszik a szoftvertervezésben és -fejlesztésben, és számos adatmodellező eszköz áll rendelkezésre, amelyek segíthetnek az adatok leírásában és dokumentálásában. Néhány példa:

ER/Studio: Az ER/Studio az egyik legnépszerűbb adatmodellező eszköz, amely számos funkcióval rendelkezik az adatok modellezéséhez és dokumentálásához. Támogatja a különböző adatbázis-kezelő rendszereket és lehetővé teszi az adatok struktúrájának leírását.

Toad Data Modeler: A Toad Data Modeler egy másik népszerű adatmodellező eszköz, amely számos funkcióval rendelkezik az adatok modellezéséhez és dokumentálásához. Támogatja az adatbázis-kezelő rendszereket, lehetővé teszi az adatok struktúrájának leírását és segíti az adatok integrációját.

PowerDesigner: A PowerDesigner egy további nagyon hatékony adatmodellező eszköz, amely számos funkcióval rendelkezik az adatok modellezéséhez és dokumentálásához. Támogatja az adatbázis-kezelő rendszereket, lehetővé teszi az adatok struktúrájának leírását, és segíti az adatok integrációját.

Oracle SQL Developer Data Modeler: Az Oracle SQL Developer Data Modeler egy további hatékony adatmodellező eszköz, amely számos funkcióval rendelkezik az adatok modellezéséhez és dokumentálásához. Támogatja az Oracle adatbázis-kezelő rendszereket, lehetővé teszi az adatok struktúrájának leírását és segíti az adatok integrációját.


sakila.mwb – MySQL Workbench
MySQL Model
EER Diagram
Navigator
Info
Catalog
Layers
User Types
sakila
Tables
actor
address
category
city
country
customer
film
film_actor
film_category
film_text
inventory
language
Description
Properties
History
Name
Value
Color: #499E21
Expanded:
Height: 258
Indices Expanded:
Left: 37
Locked:
Manual Sizing:
Name: staff
Summarize Display: -1
Top: 61
Triggers Expanded:
Width: 120
Customer data
country
customer
city
address
Customer related data
Inventory
film
language
inventory
Business
staff
store
payment
rental
Views
film_list
nicer_but_slower_film_list
actor_info
sales_by_store
sales_by_film_category
staff_list
customer_list
Film
film_in_stock
film_not_in_stock

4.6 Adatfolyam elemző eszköz

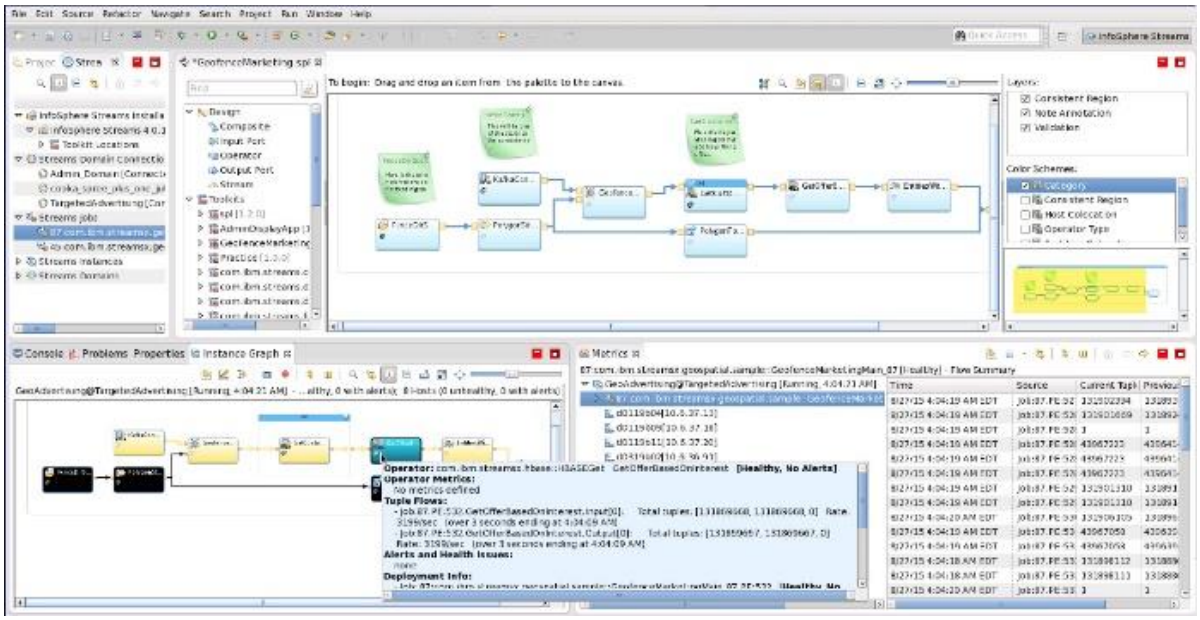
Az adatfolyam elemzése az adatok mozgásának, feldolgozásának és tárolásának elemzését jelenti az adatbázisokban és az alkalmazásokban. Számos adatfolyam elemző eszköz áll rendelkezésre az adatfolyamok elemzéséhez és optimalizálásához. Néhány példa:

Apache NiFi: Az Apache NiFi egy nyílt forráskódú, könnyen használható adatfolyam-kezelő eszköz, amely lehetővé teszi az adatok mozgatását, átalakítását és integrálását különböző források és célok között. Segítségével az adatfolyamokat a rendszer hatékonyabban és biztonságosabban kezelhetjük.

IBM InfoSphere Streams: Az IBM InfoSphere Streams egy további hatékony adatfolyam-kezelő eszköz, amely lehetővé teszi az adatfolyamok valós idejű elemzését és feldolgozását. Segítségével az adatfolyamokat nagy sebességgel és magas pontossággal lehet kezelni.

Apache Kafka: Az Apache Kafka egy másik népszerű adatfolyam-kezelő eszköz, amely lehetővé teszi az adatfolyamok feldolgozását és integrálását különböző források és célok között. Segítségével az adatfolyamokat nagy sebességgel lehet kezelni, és lehetővé teszi az adatok valós idejű elemzését.

Talend Data Streams: A Talend Data Streams egy további hatékony adatfolyam-kezelő eszköz, amely lehetővé teszi az adatfolyamok mozgatását, átalakítását és integrálását különböző források és célok között. Segítségével az adatfolyamokat hatékonyan lehet kezelni, és lehetővé teszi az adatok valós idejű elemzését és feldolgozását.



4.7 Adatfolyam modellező eszköz

Az adatfolyam modellező eszközök lehetővé teszik az adatfolyamok tervezését és modellezését, és lehetővé teszik a tervezési folyamat hatékonyabbá tételét. Néhány példa az adatfolyam modellező eszközökre:

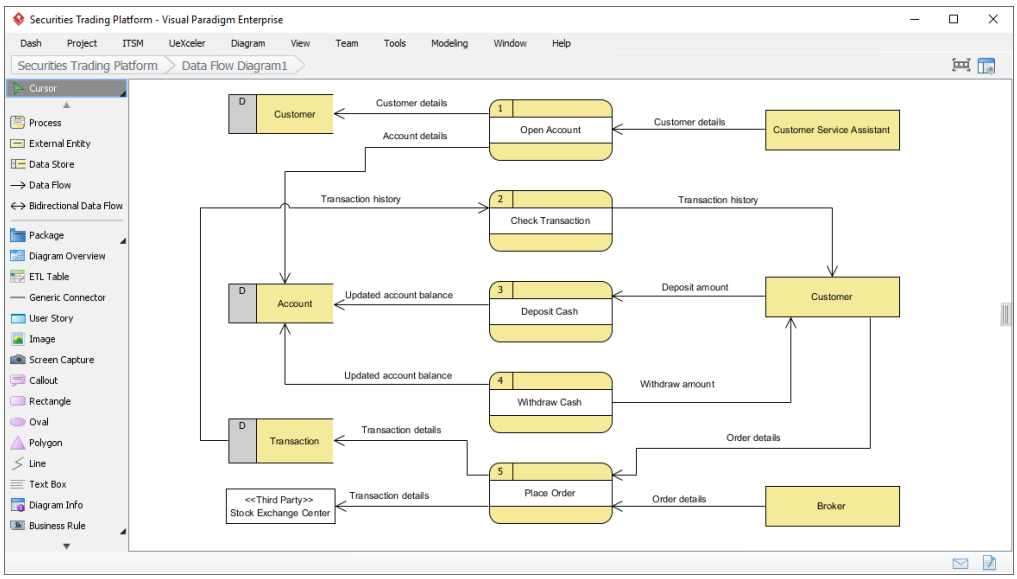
Apache NiFi: Az Apache NiFi nem csak az adatfolyam-kezelést, hanem az adatfolyam-modell tervezést is támogatja. Lehetővé teszi az adatfolyamok létrehozását és konfigurálását a rendszer különböző részein.

Altova MapForce: Az Altova MapForce egy hatékony adatátviteli eszköz, amely lehetővé teszi az adatfolyam-modellek tervezését, az adatok konvertálását és integrálását különböző források és célok között.

Visio: A Microsoft Visio egy nagyon sokoldalú diagramkészítő eszköz, amely lehetővé teszi az adatfolyam-modell tervezését és modellezését. Az adatfolyam-modell tervezését a Visio áramlási diagramok, folyamatábrák és más típusú diagramok segítségével lehet elvégezni.

Lucidchart: A Lucidchart egy további könnyen használható diagramkészítő eszköz, amely lehetővé teszi az adatfolyam-modell tervezését és modellezését.

Ezek az eszközök lehetővé teszik az adatfolyamok hatékonyabb tervezését és modellezését, ami lehetővé teszi az alkalmazások hatékonyabb működését és az adatok hatékonyabb kihasználását. A megfelelő adatfolyam modellező eszköz kiválasztása az adott projekthez és az alkalmazás igényeihez igazítva segíti a hatékonyabb szoftvertervezést és -fejlesztést.



5 Tervezési Módszertan

A tervezési módszertan egy meghatározott módszer vagy folyamat, amely segíti a szoftvertervezőket és fejlesztőket a projektterv elkészítésében és megvalósításában. A tervezési módszertanok általában egy sor lépést határoznak meg, amelyeket a szoftvertervezőnek követnie kell, hogy egy jól átgondolt és hatékony tervet készítsen.

Módszertani elemek

1. **Modellek**: az üzleti, rendszer- vagy alkalmazásmodellek használata az elképzelések megrajzolásához és a tervezési folyamat támogatásához.
2. **Nyelv és szintaxis**: az adott tervezési módszertan használja-e az UML-t vagy más nyelvet, például a BPMN-t.
3. **Lépések**: a tervezési folyamat során szükséges lépések és tevékenységek.
4. **Dokumentáció**: milyen dokumentumokat kell létrehozni a tervezési folyamat során.
5. **Szerepek**: a tervezési folyamatban résztvevő szerepek és felelősségi köreik.
6. **Eszközök**: milyen eszközöket kell használni a tervezési folyamat során, például modellalkotó eszközöket vagy diagramkészítő szoftvereket.

Módszertanok (példák)

- **Agile**: Az agilis módszer a fejlesztési folyamatot inkrementálisan végzi, és a tervezési folyamat is inkrementális és iteratív.
- **Waterfall**: A vízesés módszer egy szekvenciális modell, ahol a tervezési folyamat a projekt kezdetén történik.
- **RUP**: A RUP (Rational Unified Process) egy formális folyamatmodell, amely az UML-t használja, és a tervezési folyamatot számos iterációval végzi.
- **TOGAF**: A TOGAF (The Open Group Architecture Framework) egy kiterjedt architektúra keretrendszer, amely segíti a nagyvállalatokat az üzleti folyamatok és az IT architektúra közötti kapcsolatok szabályozásában.
- **Design Thinking**: A Design Thinking egy emberközpontú tervezési módszertan, amely hangsúlyozza az emberek igényeinek és problémáinak megértését az alkalmazás vagy rendszer tervezése során.

5.1 Enterprise Architecture - Vállalati Architektúra

A vállalkozás az egyik legösszetettebb ember alkotta rendszer, amely emberi, politikai, társadalmi, szoftveres, hardveres és technológiai összetevőkből áll. Egy számottevő méretű szervezet esetében lehetetlen, hogy egyetlen ember megértse, hogyan működnek együtt a részek, nemhogy megértse a környezetét alkotó más szervezetek rendszeréhez viszonyított helyzetét, vagy meghatározni, hogyan fejlődhet.

5.1.1 Általános felépítés

Egy vállalat átfogó architektúrája négy integrált részarchitektúrával írható le. Ezek a következők:

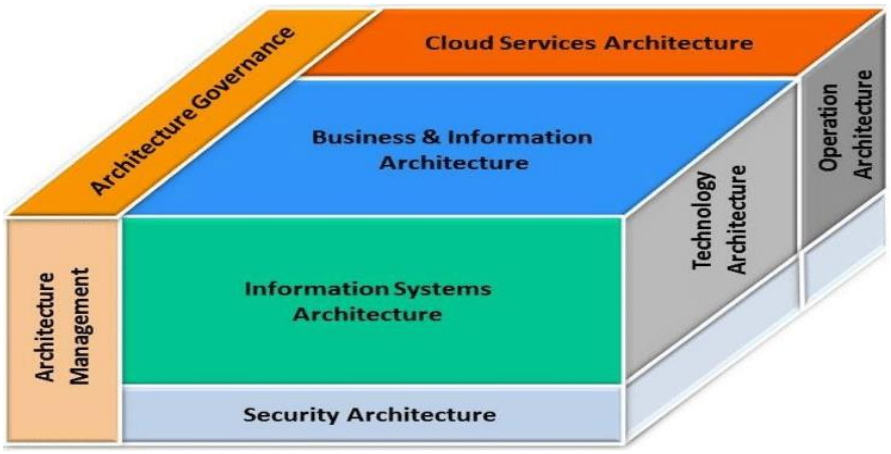
- **Üzleti architektúra** (Business Architecture)
- **Információs architektúra** (Information Architecture)
- **Alkalmazás architektúra** (Application Architecture)
- **Technológiai architektúra** (Technology Architecture)

Egy architektúra-keretrendszer tartalmazza a vállalati architektúrák létrehozásának és kezelésének eszközeit és módszereit. Ide tartoznak az architektúrák létrehozásához használt folyamatok és eszközök, az architektúra tartalmát tároló adattár, valamint a csapat szervezése, beleértve az architektúrák létrehozásának módjára vonatkozó útmutatást és az architektúrákat megvalósító csapatok irányítását.

A legtöbb keretrendszer a vállalati architektúra analóg vagy hasonló részeit írja le, mivel a felosztás nagyrészt az ezeken a területeken munkát végző szervezeti egységeken alapul. Számos egyéb architektúra létezik, amelyeket leginkább nézetekként lehetne leírni, mivel átmetszik a többi architektúrát, és a többi architektúra tartalmazza őket, de fontosságuk miatt gyakran architektúra szintjére emelik őket. Ezek közé tartoznak:

- **Biztonsági architektúra** (Security Architecture)
- **Térbeli architektúra** (Geospatial Architecture)
- **Társadalmi architektúra** (Social Architecture)

Az egyes részarchitektúrák jellemzően az Alap- és a Célarchitektúra leírásából állnak, a végrehajtható átmenetek meghatározásával, amelyeket az ütemterv ábrák (Roadmap) írnak le.



5.1.2 IT Architekt szerep hatásköre

Az IT Architekt fő feladatai közé tartozik az alkalmazások és rendszerek architektúrájának tervezése, dokumentálása és felügyelete. Az IT Architekt olyan személy, aki felelős az IT infrastruktúra tervezéséért, az üzleti célokra vonatkozó követelmények figyelembevételéért és a tervezett architektúra végrehajtásáért. Az IT Architekt feladatai a következők:

Architektúra tervezése: Az IT Architekt feladata az alkalmazások és rendszerek architektúrájának tervezése. Az IT Architekt figyelembe veszi az üzleti követelményeket és a nem funkcionális követelményeket, és olyan architektúrát tervez, amely lehetővé teszi az alkalmazások és rendszerek hatékony és megbízható működését.

Dokumentáció: Az IT Architekt feladata a tervezett architektúra dokumentálása. Az IT Architekt részletes dokumentációt készít az alkalmazások és rendszerek architektúrájáról, amely magában foglalja az architektúra leírását, az alkalmazások és rendszerek komponenseinek leírását, az adatáramlás leírását és az interfészek leírását.

Architektúra felügyelete: Az IT Architekt feladata az alkalmazások és rendszerek architektúrájának felügyelete. Az IT Architekt gondoskodik arról, hogy az alkalmazások és rendszerek az architektúra tervezése szerint működjenek, és hogy az architektúra folyamatosan fejleszthető legyen.

Követelmények figyelembevétele: Az IT Architekt feladata az üzleti követelmények és a nem funkcionális követelmények figyelembevétele. Az IT Architekt figyelembe veszi az üzleti célokat és a követelményeket, és olyan architektúrát tervez, amely lehetővé teszi a célok elérését.

Fejlesztési folyamat támogatása: Az IT Architekt feladata az alkalmazások és rendszerek fejlesztési folyamatának támogatása. Az IT Architekt biztosítja, hogy az alkalmazások és rendszerek architektúrája megfelelő legyen a fejlesztési folyamat követelményeinek, és hogy az architektúra támogassa a folyamatos integrációt és a folyamatos szállítást.

5.1.2.1 Nem IT Tervező feladatok

Az IT Architektnek nem minden esetben kell részt vennie az alkalmazások és rendszerek fejlesztési folyamatában vagy a rendszerüzemeltetésben.

Az IT Tervező nem feladata

Programozás: Az IT Architektnek általában nincs közvetlen felelőssége a kódírásban. Az IT Architekt a rendszerek architektúrájának tervezésére és felügyeletére összpontosít, és nem programozza az alkalmazásokat vagy a rendszereket.

Rendszerüzemeltetés: Az IT Architekt általában nem felelős a rendszerüzemeltetésért. Az IT Architekt feladata az alkalmazások és rendszerek architektúrájának tervezése és dokumentálása, azonban az üzemeltetési feladatok általában más személyekhez vagy csoportokhoz tartoznak.

Projektmenedzsment: Az IT Architekt feladata az alkalmazások és rendszerek architektúrájának tervezése és felügyelete, azonban az IT Architekt általában nem vesz részt a projektmenedzsmentben vagy a projekttervezésben. Az IT Architekt együttműködik a projektmenedzserekkel, azonban nem felelős a projektmenedzsment feladatainak teljesítéséért.

5.1.3 Architektúra mentesség

A nem architekturális szemlélet az alkalmazások és rendszerek tervezésének és fejlesztésének egyfajta megközelítése, amelyben az architektúra nem kap kiemelt szerepet. A nem architekturális szemléletben az alkalmazások és rendszerek tervezése és fejlesztése gyakran olyan technológiai részletekre összpontosít, mint például a programozási nyelvek, a fejlesztői környezetek és a fejlesztési módszertanok, és az architektúra tervezése későbbi szakaszban történik, vagy egyszerűen figyelmen kívül hagyják.

A nem architekturális szemlélet alapvető jellemzői

A tervezés technológiai szempontokra összpontosít: A nem architekturális szemléletben az alkalmazások és rendszerek tervezése során a tervezés gyakran technológiai szempontokra összpontosít, például a programozási nyelvekre, a fejlesztői környezetekre és a fejlesztési módszertanokra. Az architektúra tervezése nem kap kiemelt szerepet.

Az architektúra tervezése későbbi szakaszban történik: A nem architekturális szemléletben az architektúra tervezése gyakran későbbi szakaszban történik, miután a technológiai részletek már kiválasztásra kerültek. Az architektúra tervezése figyelmen kívül hagyja az alkalmazások és rendszerek funkcionális és nem funkcionális jellemzőit.

A folyamatos fejlesztés nem garantált: A nem architekturális szemléletben az alkalmazások és rendszerek folyamatos fejlesztése és javítása nem garantált. Az architektúra figyelmen kívül hagyása és a technológiai szempontokra összpontosítás gyakran olyan alkalmazások és rendszerek létrehozásához vezet, amelyek nehézkesek és megbízhatatlanok, és amelyek nem könnyen karbantarthatók vagy fejleszthetők.

A nem architekturális szemlélet lehetővé teszi a tervezők számára, hogy gyorsan fejlesszenek és létrehozzanak alkalmazásokat és rendszereket, azonban az alkalmazások és rendszerek megbízhatósága, hatékonysága és karbantarthatósága sok esetben problémás lehet. Az architektúra tervezése és az architekturális szemlé

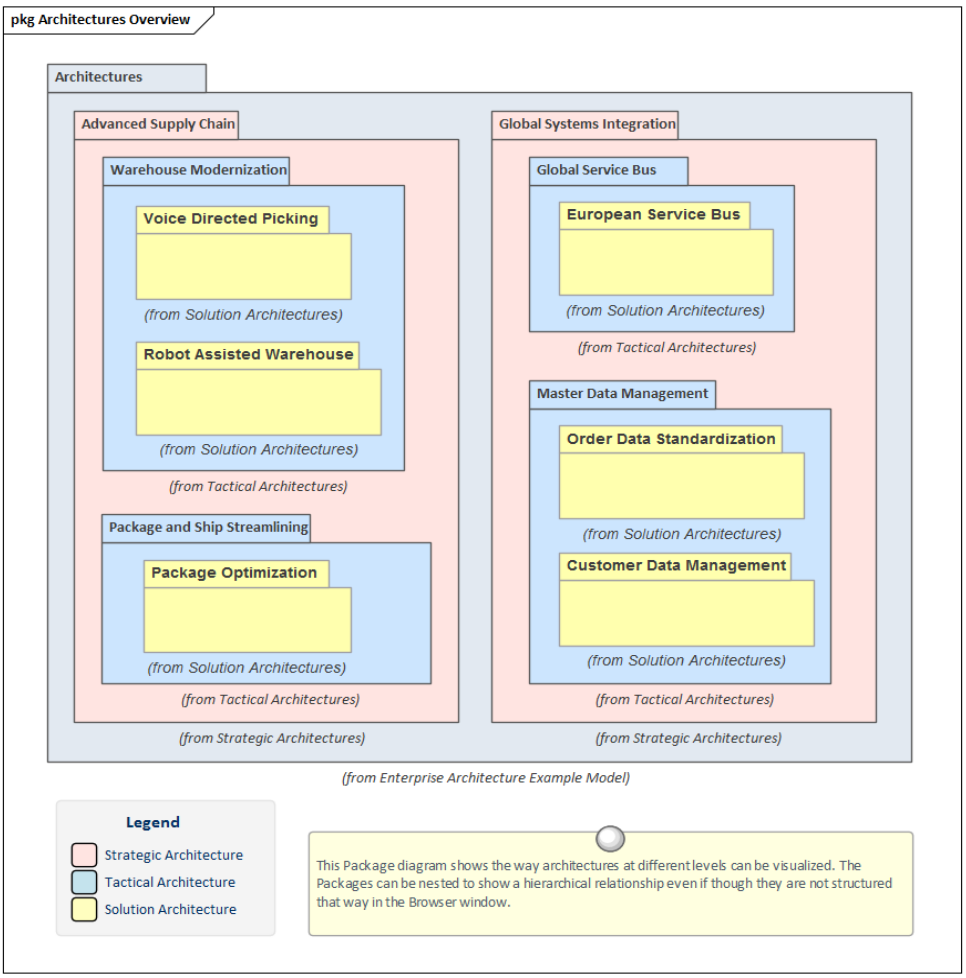
5.1.4 Architektúra Szintek

Egy vállalat összetett és jellemzően hierarchikus struktúrával rendelkezik, és a struktúra egyes szintjein kell architektúrákat létrehozni. Az architektúrák e hierarchiája analóg a célok és képességek hierarchiájával, és intuitív módon illeszkedik a stratégiai, program- és projektszintű felosztáshoz.

Egy kis szervezetben lehetséges lehet egyetlen olyan architektúra létrehozása, amely lefedi a stratégiai szintet és a projekt- vagy képességszinteket, de egy számottevő méretű vállalatnál általában legalább három különálló szintre van szükség.

A szintek elnevezése közmegegyezéssel (de facto) a **The Open Group Architecture Framework (TOGAF)** módszertanát követi.

- **Stratégiai** - Hosszú távú, 3-5 év közötti időszak
- **Taktikai** - középtávú, 1-2 év közötti időszak
- **Megoldás** - Rövid távú, 6-12 hónapos időtartamú



5.1.5 Architektúra Minták

Az architektúra minták az architektúra tervezési folyamatában fontos szerepet játszanak, mert olyan ismétlődő tervezési sablonok, amelyek bizonyítottan hatékonyak az alkalmazások és rendszerek tervezése során.

5.1.5.1 Előnyök

Időtakarékosság: Használatuk lehetővé teszi az architektúra tervezésének gyorsabb és hatékonyabb folyamatát. Biztosítják a tervezési sablonokat és keretrendszereket, amelyek segítenek a tervezési folyamat felgyorsításában.

Bizonyított hatékonyság: Az alkalmazások és rendszerek széles körében alkalmazhatók, és bizonyítottan segítenek a hatékonyabb és megbízhatóbb architektúra kialakításában.

Minőségi tervezés: Biztosítják az architektúra minőségét és megbízhatóságát azzal, hogy segítenek a tervezőknek az architektúra nem funkcionális jellemzőinek figyelembevételében, és lehetővé teszik az alkalmazások és rendszerek hatékonyabb működését.

Könnyű karbantarthatóság: Biztosítják a megfelelő szerveződést és moduláris felépítést, amely lehetővé teszi az alkalmazások és rendszerek könnyebb karbantartását és fejlesztését.

Közös nyelv: Segítenek a tervezőknek közös nyelvet kialakítani az architektúra tervezése során.

5.1.5.2 Tervezési sablonok

Az architektúra sablonok általában az alkalmazás vagy rendszer különböző részeit írják le, mint például az adatbázis kezelés, a felhasználói felület tervezése vagy az adatfeldolgozás.

Model-View-Controller (MVC): Az MVC egy olyan architektúra minta, amely különválasztja az alkalmazás adatmodelljét, a felhasználói felületet és az üzleti logikát. Az MVC architektúra minta javítja az alkalmazás karbantarthatóságát és tesztelhetőségét.

Repository minta: A Repository minta egy olyan architektúra minta, amely az adatelérési rétegben segít. Ez az architektúra minta segít az adatbázis kommunikáció kódjának elválasztásában az alkalmazás üzleti logikájától.

Pipe and Filter minta: A Pipe and Filter minta egy olyan architektúra minta, amely egy adatfolyamot szegmentál kisebb, független részekre, és minden rész feladata az adatok feldolgozása. Ez az architektúra minta javítja a szoftver modularitását, skálázhatóságát és újrafelhasználhatóságát.

Publish-Subscribe minta: A Publish-Subscribe minta egy olyan architektúra minta, amely az esemény vezérelt architektúrában segít. Ez az architektúra minta segíti az alkalmazások és rendszerek aszinkron kommunikációját és javítja az alkalmazások teljesítményét és skálázhatóságát.

Az architektúra mintázatok használata javítja a szoftvertervezés hatékonyságát, javítja az alkalmazások minőségét, csökkenti a hibákat és javítja a karbantarthatóságot és újra felhasználhatóságot.

5.1.6 Architektúra-alapelvek (Architecture Principles)

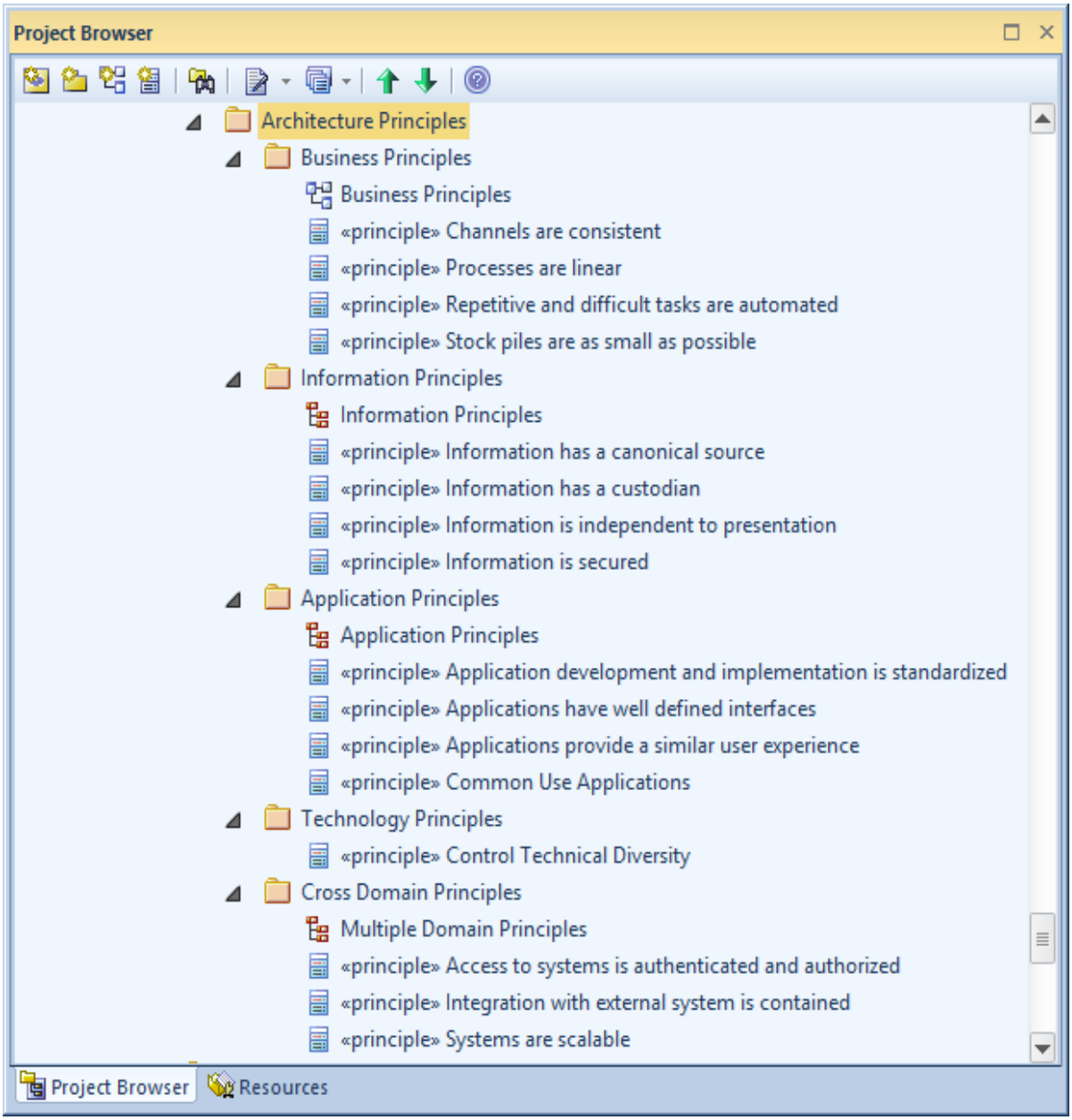
Az Architektúra-elvek kritikus szerepet játszanak az architekturális munka irányításában (**Governance**), amelynek végső soron az a feladata, hogy meghatározza a vállalat jövőbeli irányát és a jövőbeli állapot (**Target Architecture**) eléréséhez szükséges átmeneteket (**Transition**).

Az Architektúra-elveket általában az Architektúra Program létrehozásakor hozzák létre, majd az Architektúra Irányító Bizottság (**Architecture Steering Committee**) felülvizsgálja és ratifikálja őket. Fontos, hogy összehangolódjanak a meglévő vállalati alapelvekkel, de az is fontos, hogy ezeket úgy kezeljék, hogy azok értelmezhetőek és alkalmazhatóak legyenek az architektúra szintjén. Gyakori, hogy az egyes architektúra-területekhez (**Architecture Domains**) egy-egy alapelv-készletet határoznak meg, egy közös csoporttal, amely több területre is kiterjed.

5.1.6.1 Alapelv-csoportok

- **Üzleti alapelvek**
- **Információs (adat) alapelvek**
- **Alkalmazási elvek**
- **Technológiai alapelvek**
- **Biztonsági alapelvek**
- **Több területet átfogó elvek**

Az elveket az architektúra és az azt felhasználó Végrehajtási Kezdeményezések (**Implementation Initiatives**) kontextusában kell alkalmazni. A legjobb gyakorlat az, ha a Területi Architektek (**Domain Architects**) a megvalósítási csapatokkal együttműködve gondosan meghatározzák, hogy mely elvek vonatkoznak egy adott kezdeményezésre, és hogyan kell azokat értelmezni és alkalmazni.



Architektúra alapelvek

5.1.6.2 Rendelkezések

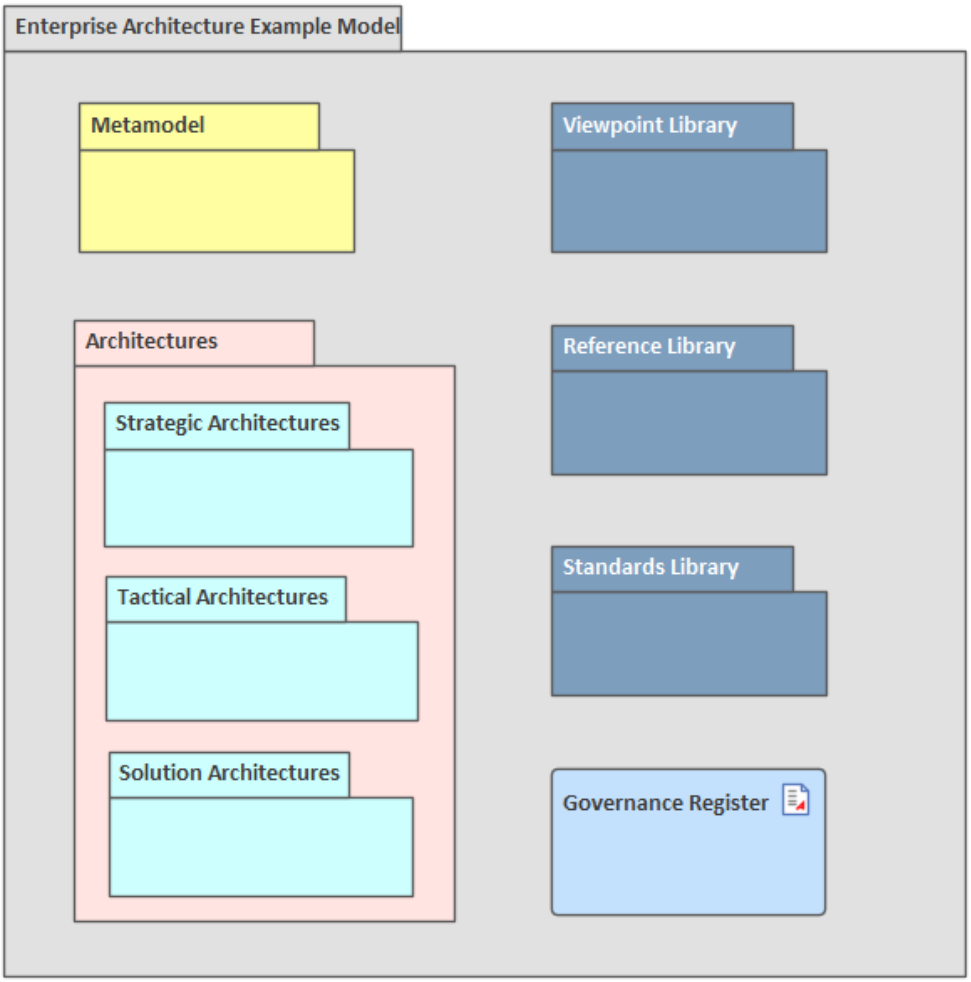
Lesznek olyan helyzetek, amikor egy vagy több alapelv alkalmazása hátrányosan befolyásolná a kívánt eredményeket. Ezekben a helyzetekben a Fő Architekt (**Chief Architect**) mentesítést adhat ki, amely felmenti a megvalósító csoportot az elv betartása alól. Ezeket a felmentéseket az Irányítási Nyilvántartásban (**Governance Register**) kell rögzíteni.

5.1.7 Tervezés

A vállalati architektúra felhasználható a szervezet különböző absztrakciós szinteken történő vizualizálására és olyan ütemtervek készítésére, amelyek megmutatják, hogyan lehet a szervezetet az alapállapotból (jelenlegi állapotból – **Baseline Architecture**) egy célállapotba (jövőbeli állapotba – **Target Architecture**) átállítani.

5.1.7.1 Tervezési környezet

Az architektúra tervezési környezet felfogható egy „architektúra-tárként”, amely magában foglalja a fontos architekturális inputokat és outputokat, beleértve magukat az architektúrákat (**Architectures**), az elemeket, amelyekből összeállnak (**Metamodel**), szabványokat (**Standards Library**), hivatkozásokat (**Reference Library**), elveket és az Irányítási Nyilvántartást (**Governance Register**).

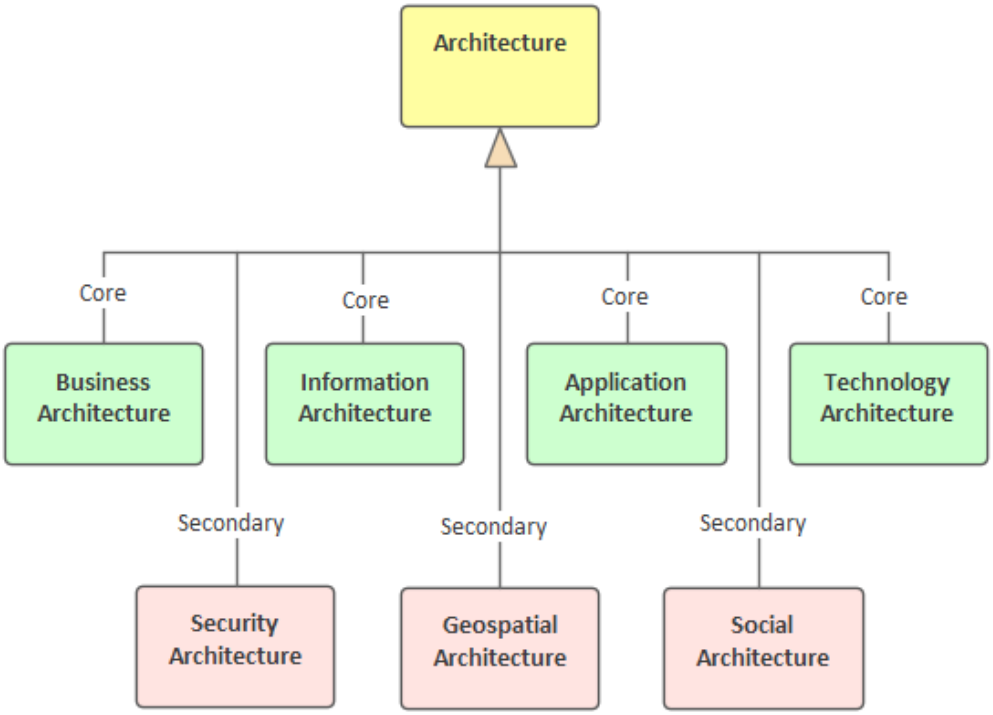


5.1.7.2 Architektúra Metamodell

A metamodell egy modell modellje, amely leírja az architektúrák felépítéséhez felhasználható elemeket és kapcsolatokat. **A metamodell nyelvtanként funkcionál**, amely meghatározza a típusokat és azt, hogy azok hogyan kapcsolódhatnak egymáshoz.

Nagyon fontos, hogy **a metamodellt már az építészeti modellek létrehozása előtt definiáljuk**, mivel ez tájékoztatja az építészeket arról, hogy milyen elemeket kell használniuk, és hogyan használhatók együtt.

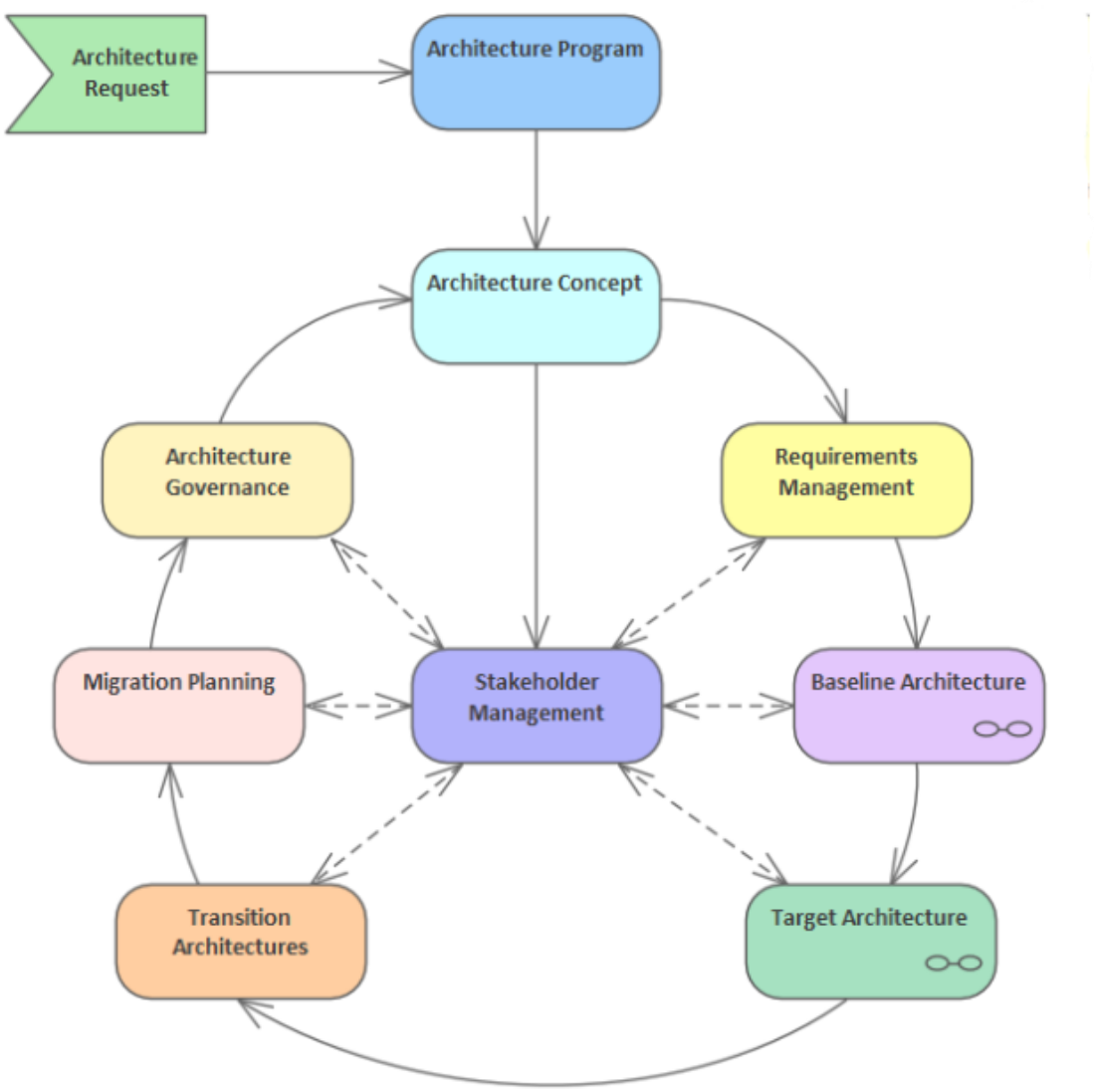
A metamodell ebben a formátumban **passzív modell, és csak útmutatóként és kommunikációs eszközként szolgál**; ha formálisabb modellre van szükség, akkor UML-profil készíthető.



5.1.7.3 Architektúra tervezési folyamat

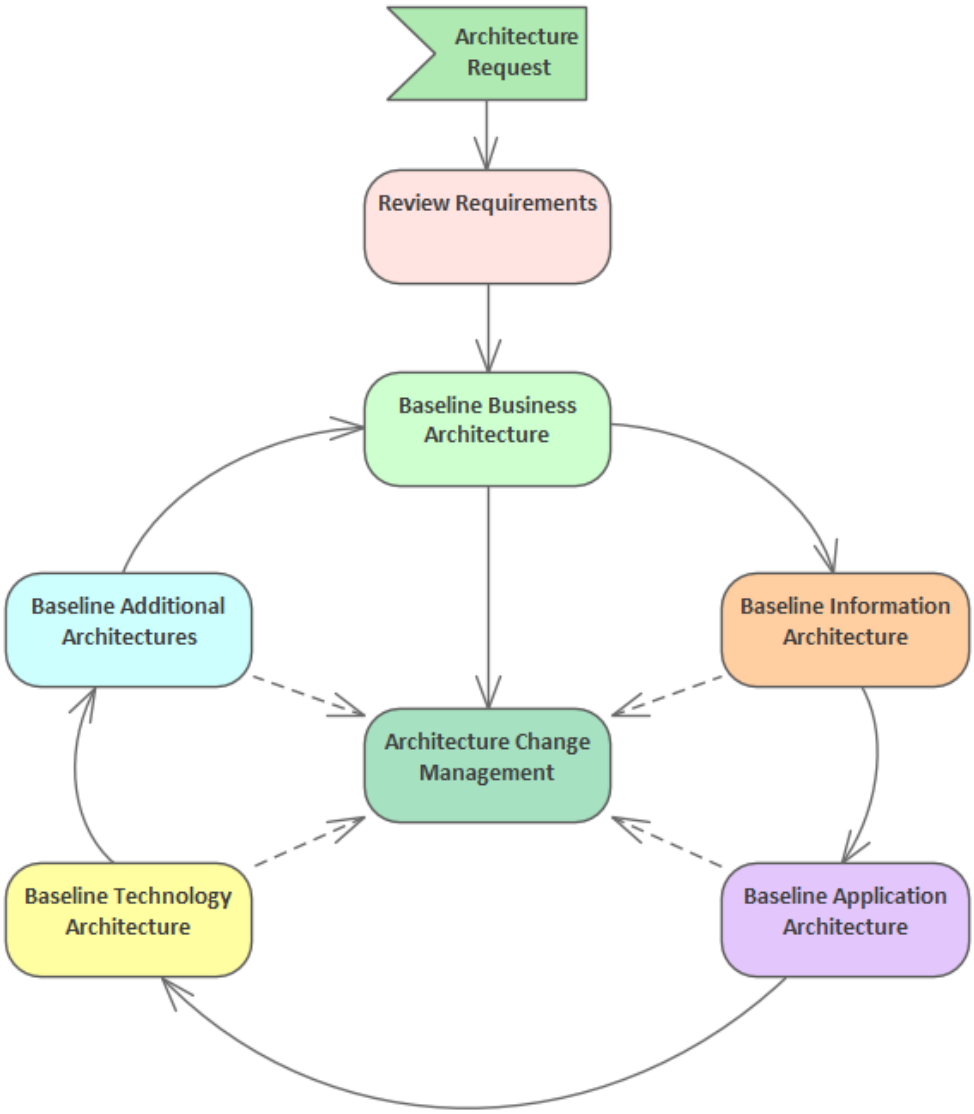
Az architektúrák fejlesztésének módját egy architektúra-folyamat vagy -módszer írja elő. Nem minden Architektúra Keretrendszer (Architecture Framework) biztosít meghatározott (munka)folyamatot, így egy szervezetnek magának kell megalkotnia és konfigurálnia a saját tervezési/építési folyamatát.

Architektúra fejlesztési folyamat (Architecture Development Process)



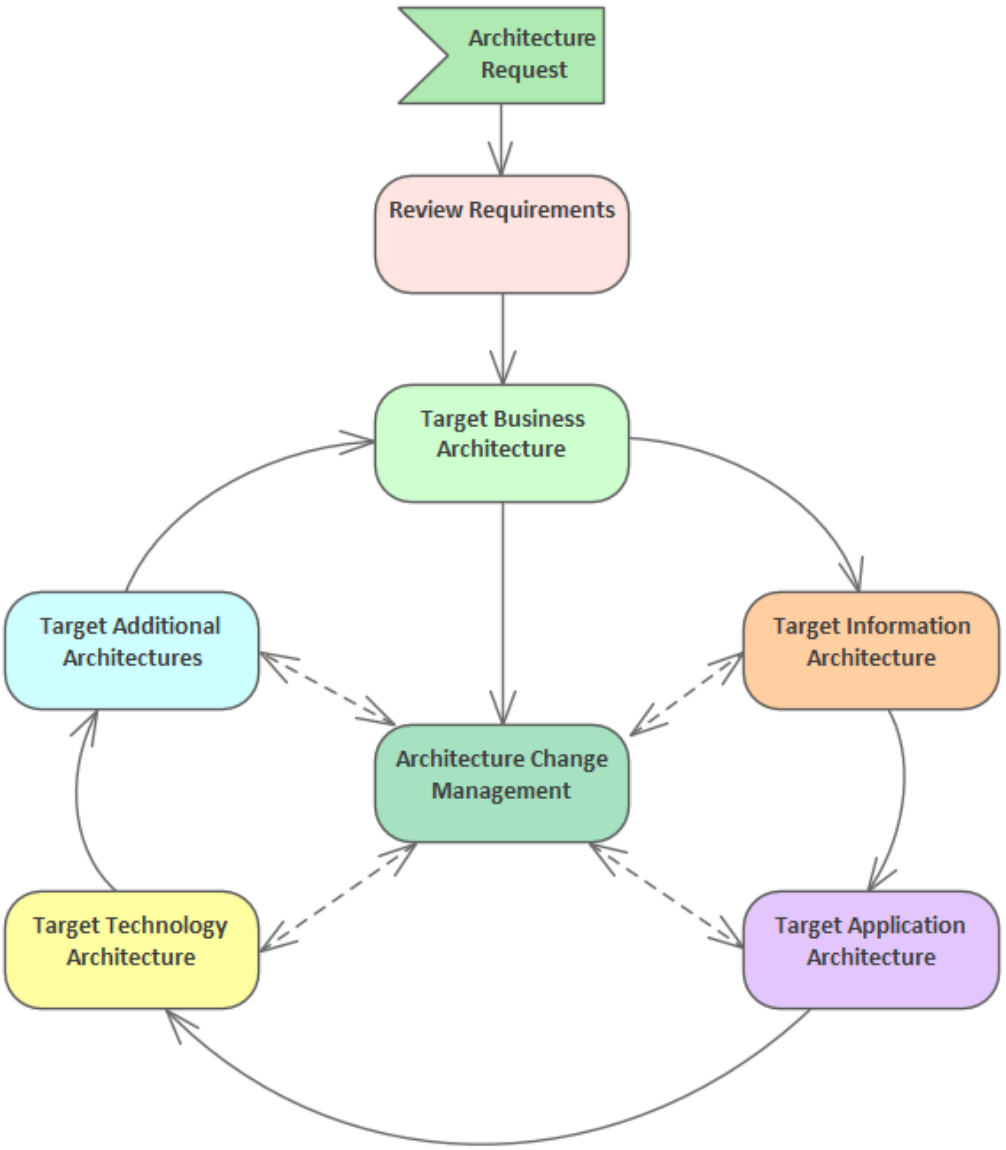
Alaparchitektúra (Baseline Architecture)

Az alaparchitektúrák fontossága az, hogy megteremtsék azt a kiindulópontot, amely lehetővé teszi a célarchitektúrákra való átmenet meghatározását. Gyakran előfordul, hogy léteznek olyan dokumentációk és modellek, amelyekből anyagot lehet gyűjteni az alapszintű tároló feltöltéséhez. Például a legtöbb szervezetnek volt már legalább néhány kísérlete a meglévő folyamatok modellezésére, talán egy üzleti újra tervezési erőfeszítés részeként, és gyakran létezik egy vagy több információs modell és hardverdiagram.



Célarchitektúra (Target Architecture)

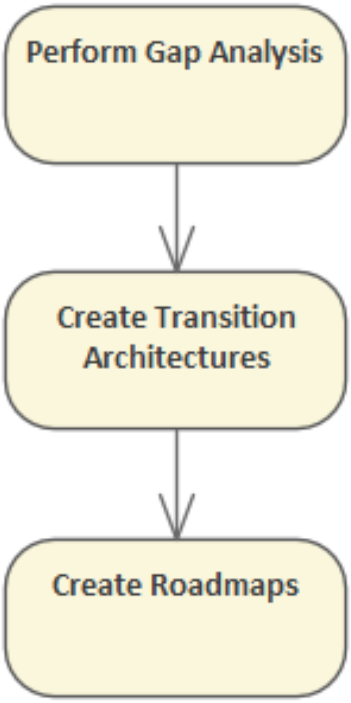
Ezek határozzák meg azokat az architektúrákat, amelyek megvalósítják az üzleti stratégiákat és értéket biztosítanak a szervezet számára. Miután ezek ismertek és az **Alaparchitektúrák** eléggé kidolgozottak, az architekt nekiláthat az **Átmeneti Architektúrák** meghatározásának és az ütemtervek (**Roadmap**) elkészítésének, amelyek előírják, hogy a **Cél Architektúrát** hogyan lehet a gyakorlatban átmeneti lépésekkel elérni.



Átmenet architektúra (Transition Architecture)

Az **Alaparchitektúra** és a végső **Cél Architektúra** közötti lépcsőfokok, és elméletileg maguk is Célarchitektúrák. A jelenlegi állapotból a jövőbeni állapotba való eljutás gyakorlati lépéseit jelentik, és gyakran egy átfogó projekt projektjeiként vagy fázisaiként jelennek meg a megvalósítás szintjén.

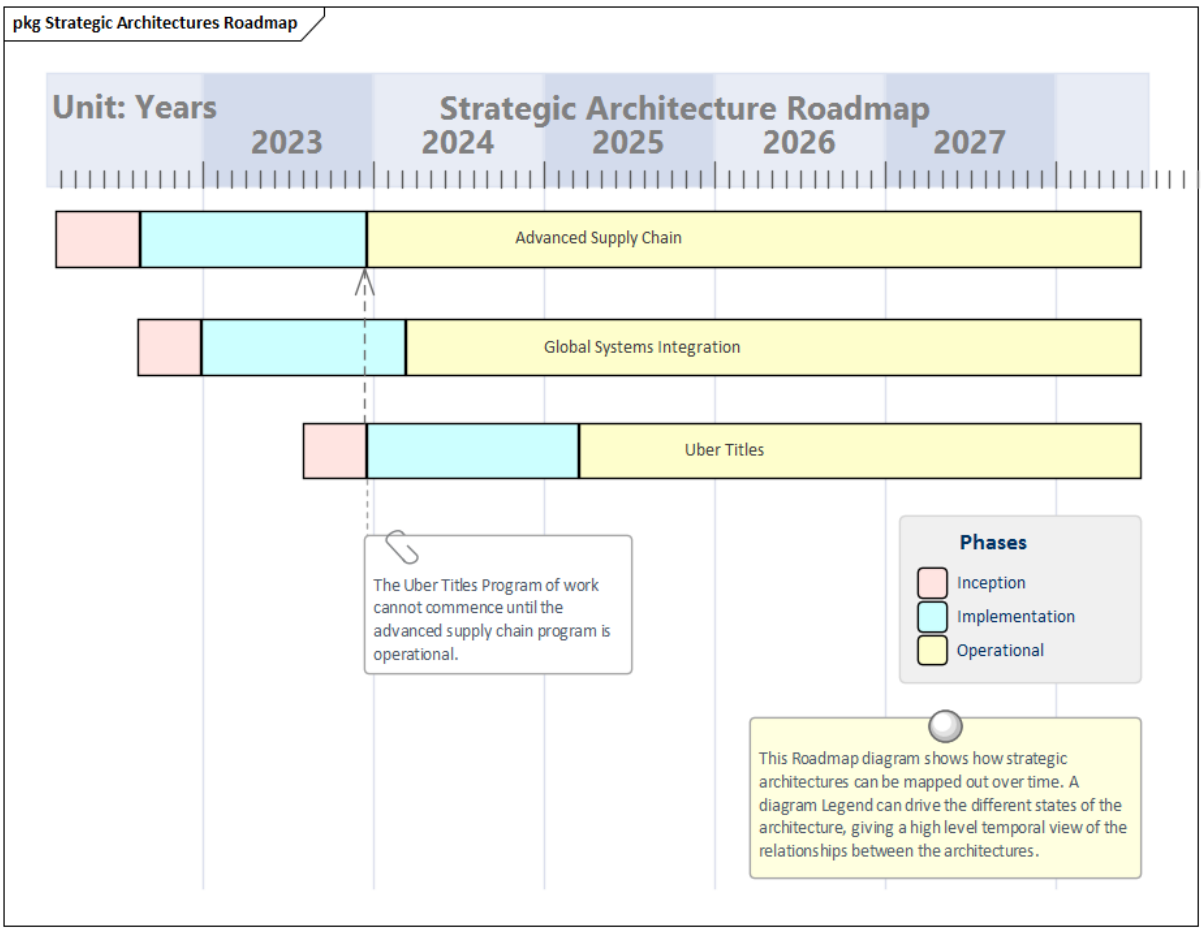
Az **Átmeneti Architektúrákhoz** kapcsolható **ütemtervek** sorozata, amik segítenek az egyik állapotból a következőbe való átmeneteket vizualizálni és megtervezni. Bármennyi ütemterv definiálható, és ezek az összes **Architektúra Tartományra** használhatók, így például lehetnek képesség-, alkalmazás- és technológiai ütemtervek.



5.1.8 Architektúra változás ütemterv (Roadmap)

Az Ütemterv-diagramok segítenek a meglévő elemek feltérképezésében az idővonalhoz képest. Ez hasznos egy rendszer időbeli változásának vizualizálásához, a két csapat közötti felelősség átcsoportosításának megtervezéséhez, több platform egyidejű támogatásának dokumentálásához vagy szinte bármilyen időalapú adat ábrázolásához. Annak érdekében, hogy a modellező és a néző mindig megfelelő kontextussal rendelkezzen, az idővonal mindig megjelenik a képernyőn, bármely mentett képen és nyomtatáskor is.

Az egyes elemek fázis- vagy állapotváltozásait az Ütemterv-diagramokon az egyes állapotokat jelképező színes szegmensekre osztott sávok speciális jelölésével lehet ábrázolni. A rendelkezésre álló állapotokat, azok megjelenési sorrendjét és a használt színeket a diagramlegenda vezérli. Minden megjelenített elem ezután megmutatja az egyes szegmensek hosszát (és így az egyes állapotok időtartamát), és lehetőséget biztosít az egyes szegmensek elrejtésére is az adott elemen.



5.1.9 Architektúra minősítés

Ahhoz, hogy egy Architektúra hatékony legyen, számos tulajdonsággal vagy jellemzővel kell rendelkeznie.

Minőség	Leírás
Erős	Az architektúrának erősnek kell lennie, és nem lehet sebezhető az üzleti, információs, alkalmazási és technológiai rendszerekben bekövetkező kisebb változásokkal szemben.
Megvalósítható	Egy olyan architektúra, amelyet nem lehet megvalósítani, azt jelenti, hogy a vállalkozás céljai és célkitűzései nem teljesülnek.
Hasznos	Az architektúrának olyan hasznossággal kell rendelkeznie, amely a megvalósítás során gyakorlati eredményeket eredményez.
Tartós	Az architektúra egy élő entitás, amely egy célállapotot ír le, és - amint megvalósul - az új alapállapot lesz. Az architektúráknak az idő múlásával tartósnak kell bizonyulniuk, és rugalmasnak kell lenniük az üzleti és műszaki környezet változásaira, amelyek az architektúrák élettartama alatt bekövetkezhetnek.
Rugalmas	Az architektúráknak rugalmasnak kell lenniük, alkalmazkodniuk kell a változó körülményekhez, és elegendő útmutatást kell nyújtaniuk a szakterületükön jártas megvalósító csapatok.
Ellenőrizhető	Lehetővé kell tenni annak ellenőrzését, hogy az architektúra a tervezett módon fog működni, és hogy az architektúrának és a vállalat azon részeinek, amelyekre hatással van, nem lesznek mellékhatásai.

Minőség	Leírás
Elegáns	Egy jól megtervezett építészet általában elegáns és egyszerű formájú, ami nyilvánvaló lesz azok számára, akik időt szánnak a tanulmányozására.
Visszakövethető	Az architektúra a vállalkozás egy adott részletességű leírása, amely nem elszigetelten létezik, hanem jellemzően kapcsolódik az üzleti ösztönzőkhöz és célokhoz, valamint más, azonos vagy magasabb vagy alacsonyabb szintű architektúrákhoz, illetve a megvalósítási programokhoz és projektekhez.

5.1.9.1 Minőség mérés

Az architektúra minősége mérhető a funkcionális és nem funkcionális jellemzők alapján. Az architektúra minőségét értékelni lehet a következő módon:

Teljesítménytesztelés: Az alkalmazások és rendszerek teljesítményét lehet mérni és értékelni különböző tesztek és metrikák segítségével. A teljesítménytesztelés segít azonosítani a teljesítményproblémákat, és lehetővé teszi azok javítását.

Kódminőség ellenőrzése: Az alkalmazások és rendszerek kódminőségét ellenőrizni lehet különböző eszközökkel, például a kódanalizátorokkal, amelyek segítenek azonosítani a kódhibákat, a redundanciákat, a sebezhetőségeket és az egyéb problémákat.

Biztonsági tesztelés: Az alkalmazások és rendszerek biztonsági tesztelésével lehet értékelni az architektúra biztonságát. A biztonsági tesztelés során különböző támadásokat szimulálnak az alkalmazások és rendszerek ellen, és tesztelik azok biztonsági védelmét.

Funkcionális tesztelés: Az alkalmazások és rendszerek funkcionális tesztelésével lehet értékelni az architektúra funkcionális jellemzőit. A funkcionális tesztelés során tesztelik az alkalmazások és rendszerek funkcionalitását és megfelelőségét az üzleti követelményekhez.

Felhasználói visszajelzések: Az alkalmazások és rendszerek felhasználói visszajelzéseiből lehet értékelni az architektúra minőségét. A felhasználói visszajelzések segítenek azonosítani az alkalmazások és rendszerek használhatósági problémáit, valamint azokat az igényeket, amelyekre az architektúrának reagálnia kell.

Az architektúra minőségének méréséhez használható továbbá a nemzetközileg elismert CMMI (**Capability Maturity Model Integration**) modell, amely lehetővé teszi a szervezetek architektúra fejlettségi szintjének értékelését és javítását. Az architektúra minőségének értékeléséhez számos eszköz, módszer és keretrendszer áll rendelkezésre, amelyek segítenek a szervezeteknek az architektúra minőségének folyamatos javításában.

5.2 Alkalmazás Architektúra

Az alkalmazás rétegeinek és azok kapcsolatainak szervezési módját írja le.

5.2.1 Alkalmazás architektúra réteg

1. **Felhasználói felület réteg**: Ez a réteg tartalmazza a felhasználói felületet és a felhasználói interakciókat kezelő komponenseket.

2. **Alkalmazási réteg**: Ez a réteg tartalmazza az üzleti logikát és az alkalmazási szolgáltatásokat, amelyek segítik a felhasználók igényeinek kielégítését.

3. **Adatkapcsolati réteg**: Ez a réteg kapcsolatot teremt az alkalmazás és az adatbázis között.

4. **Adattárolási réteg**: Ez a réteg tartalmazza az adatbázist és az adatokat, amelyeket az alkalmazás kezel.

Az alkalmazás architektúrában lévő rétegeknek az a célja, hogy szétválasszák az alkalmazás különböző részeit, és lehetővé tegyék a részek független fejlesztését, karbantartását és cseréjét. Az alkalmazás architektúra rétegei hatékonyabbá és rugalmasabbá teszik az alkalmazás kezelését és fejlesztését.

5.2.2 Alkalmazás Architektúra szint

Komponens architektúra

Az alkalmazás architektúra egyik szintje, amely az egyes komponensek, modulok és funkciók közötti kapcsolatokat és függőségeket határozza meg.

Adat architektúra

Az alkalmazás architektúra másik szintje, amely az alkalmazások által használt adatok struktúráját és az adatok áramlását tervezi meg.

Integrációs architektúra

Az alkalmazás architektúra harmadik szintje, amely az egyes alkalmazások közötti integrációt tervezi meg, beleértve a szolgáltatásokat és az interfészeket.

5.2.3 Alkalmazás Katalógus

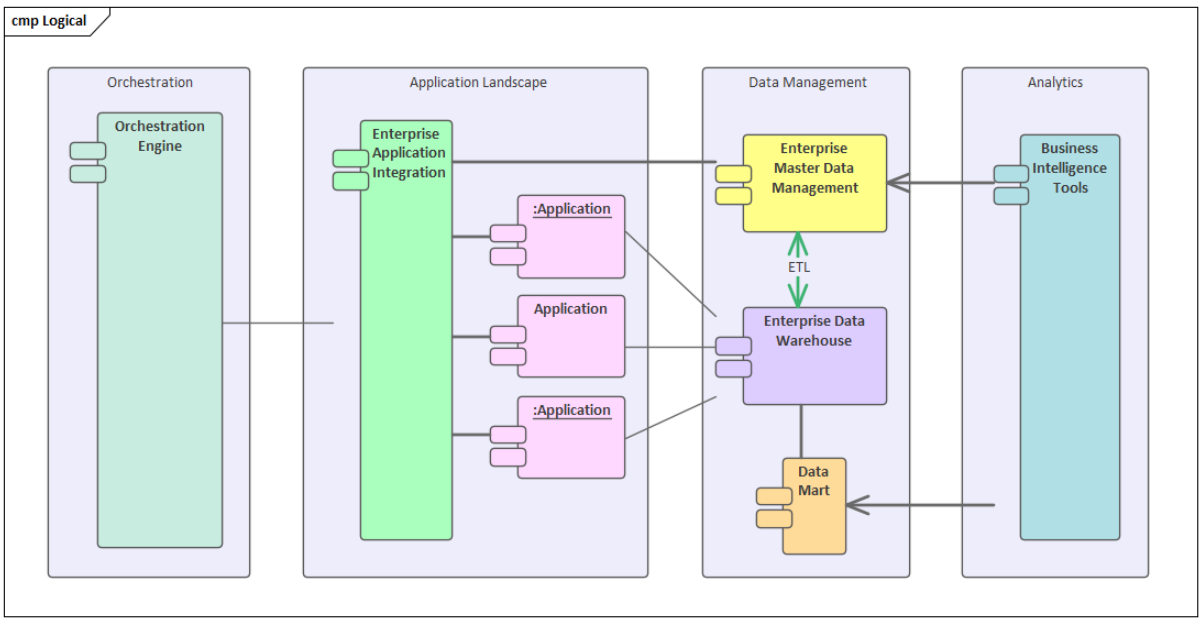
Nyilvántartja egy szervezet alkalmazásainak (Application) az információk átalakítása, továbbítása és tárolása érdekében végzett munkáját, jellemzőit.

Leírja továbbá az alkalmazások által megkövetelt vagy biztosított interfészeket, valamint azt, hogy az alkalmazások hogyan lépnek kölcsönhatásba egymással az üzleti modellekben, például az üzleti folyamatábrákban leírt tevékenységek elvégzése érdekében.

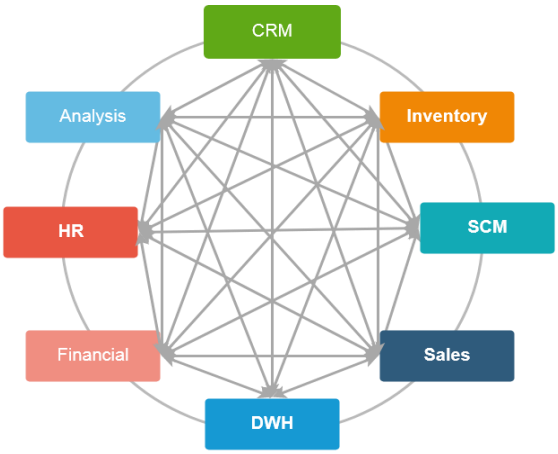
Az alkalmazások katalógusát, az interfészeket, valamint a kölcsönhatásukat leíró diagramokat és mátrixokat csak egyszer kell meghatározni vállalati szinten.

Az alkalmazás-architekt képes lesz a meglévő artefaktumok e leltárára támaszkodni új architektúrák létrehozásához, az alap- és potenciálisan a jövőbeli állapot-architektúra részeként besorolva azokat.

Ha egy architektúra új alkalmazásokat vezet be, ezeket hozzá lehet adni a célállapot leírásához.

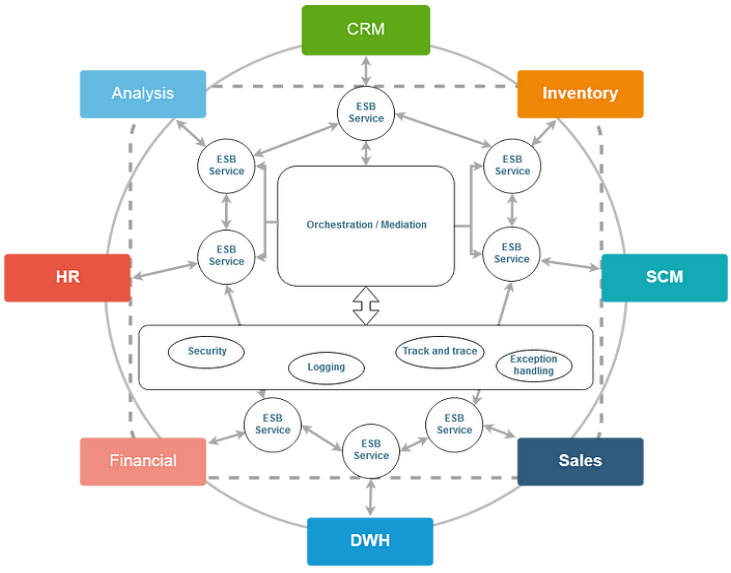


5.2.4 Alkalmazás Architektúra - Integrációs Minták



Point-to-point Integration

Hub-and-spoke Integration



ESB Integration

5.2.5 Alkalmazás Architektúra - Web Application

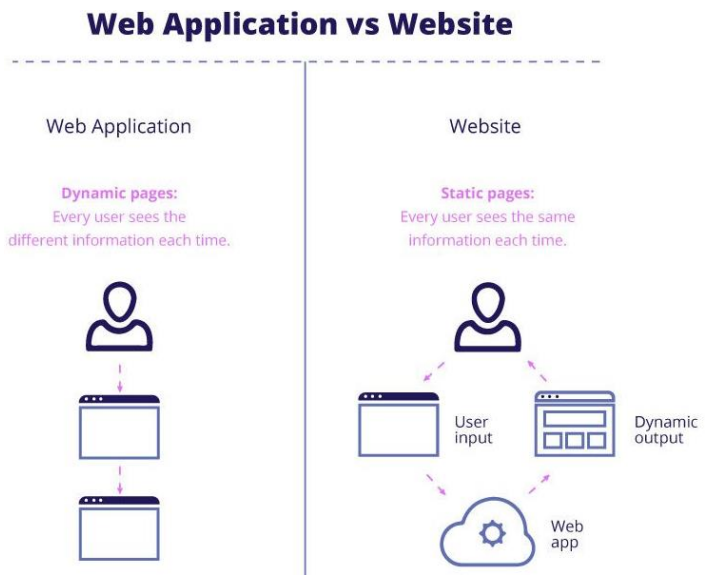
Kliens-Szerver (Client – Server) alkalmazás, aminek elemei egy böngésző (Client („Web-Agent")) és egy Webkiszolgáló ((Web) Server) között oszlanak meg.

A Webalkalmazás logikája elosztott a Szerver és a Kliens komponensek között, ahol egy csatorna biztosítja az információcserét, valamint az adattárolás helyileg („On-Premise"), vagy a (számítási) Felhőben (Cloud) található.

A Webalkalmazások a Weboldalak (Website) fejlődési szakaszaként jelentek meg, és sok közös vonásuk van. A Weboldalakat a Webalkalmazásoktól megkülönböztető tényezők az

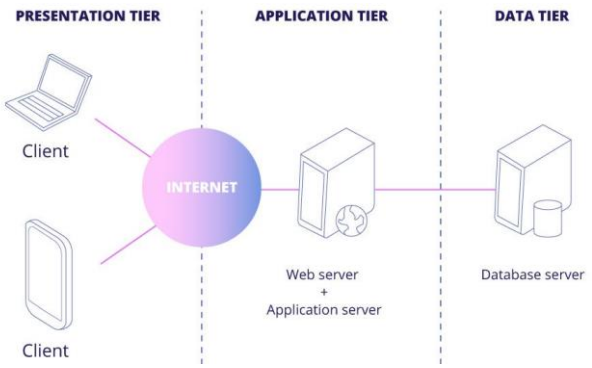
- interaktivitás
- integráció
- hitelesítés

Minél inkább testre szabható, interaktív és funkcionális egy Weboldal, annál inkább tekinthető Webalkalmazásnak.



A modern Webes alkalmazások (még mindig) a 3 szintes architektúra koncepcióját használják, amely az alkalmazásokat **prezentációs szintre**, **alkalmazási szintre** és **adatszintre** osztja.

Ebben az architektúrában minden réteg saját infrastruktúrán fut, és különböző fejlesztői munkacsoportok párhuzamosan fejleszthetik őket. Ez a struktúra lehetővé teszi az egyes szintek szükség szerinti frissítését és skálázását anélkül, hogy ez hatással lenne a többi szintre.



5.2.5.1 Webalkalmazás architektúra típusok

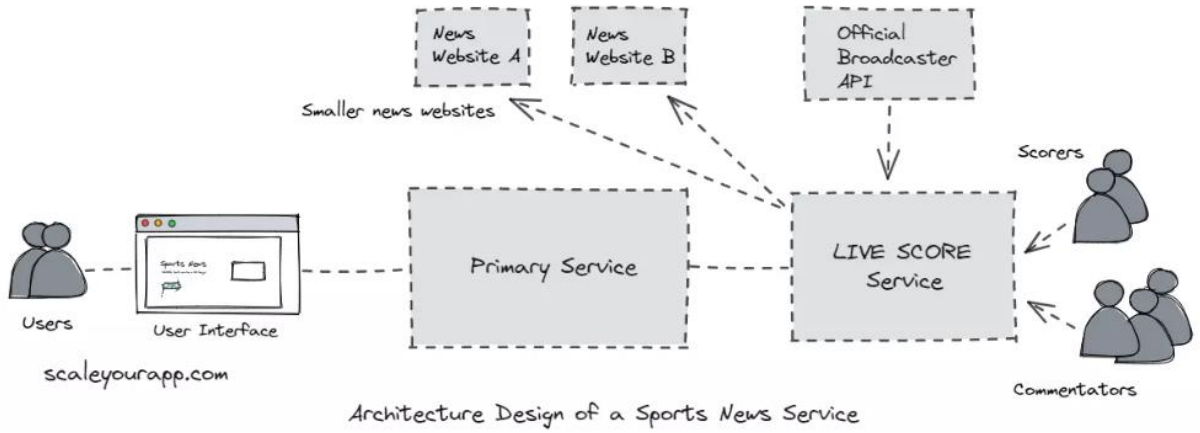
A típusok összehasonlíthatósága leíró jellemzők alapján lehetséges, mivel igen összetett rendszerekről van szó.

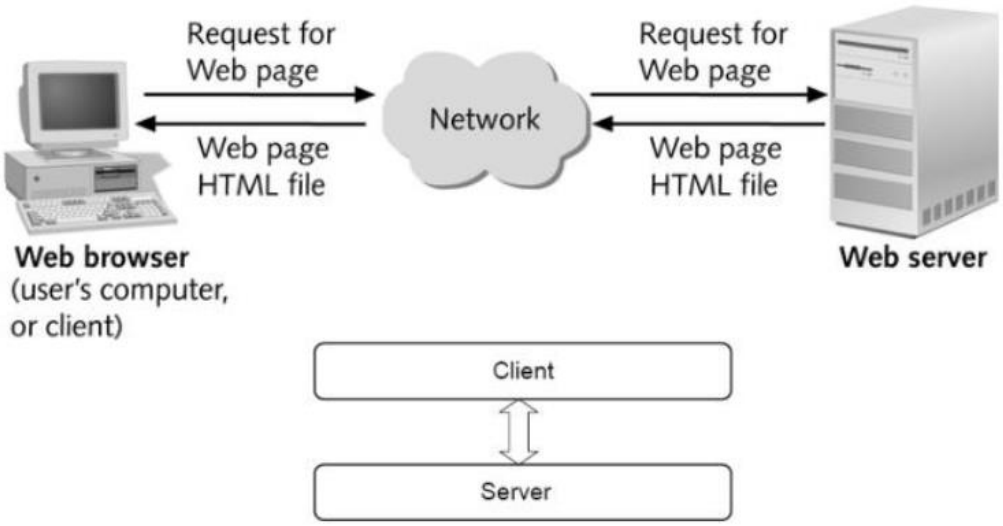
Osztályozási kategóriák

- teljesítmény
- felhasználói felület
- SEO
- linkelhetőség
- megvalósítás sebessége a fejlesztési oldalon

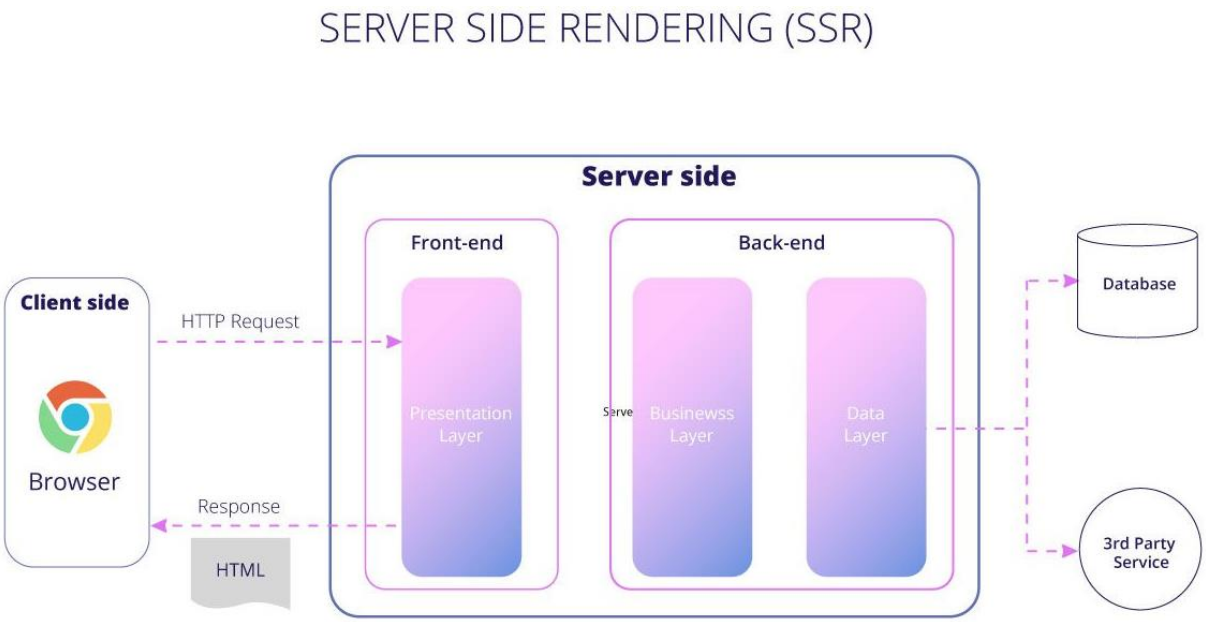
5.2.5.2 Webalkalmazás architektúra koncepció

Ha a webes alapelvekről beszélünk, akkor általában a kliens-szerver architektúrára gondolunk. Az ügyfél tartalmat kér egy szervertől, ahol az üzleti logika és az adatbázis található. Egy egyszerű JavaScript segítségével egy statikus weboldal elküldi a kérést egy szolgáltatásnak (esetleg egy API-nak). A szolgáltatás visszaküldi az adatokat, és egy HTML-oldalt jelenít meg az ügyfélnek.





5.2.5.3 Kiszolgáló oldali renderelés (SSR)



Ha a webes alapelvekről beszélünk, akkor általában a kliens-szerver architektúrára gondolunk. Az ügyfél tartalmat kér egy szervertől, ahol az üzleti logika és az adatbázis található. Egy egyszerű JavaScript segítségével egy statikus weboldal elküldi a kérést egy szolgáltatásnak (esetleg egy API-nak). A szolgáltatás visszaküldi az adatokat, és egy HTML-oldalt jelenít meg az ügyfélnek.

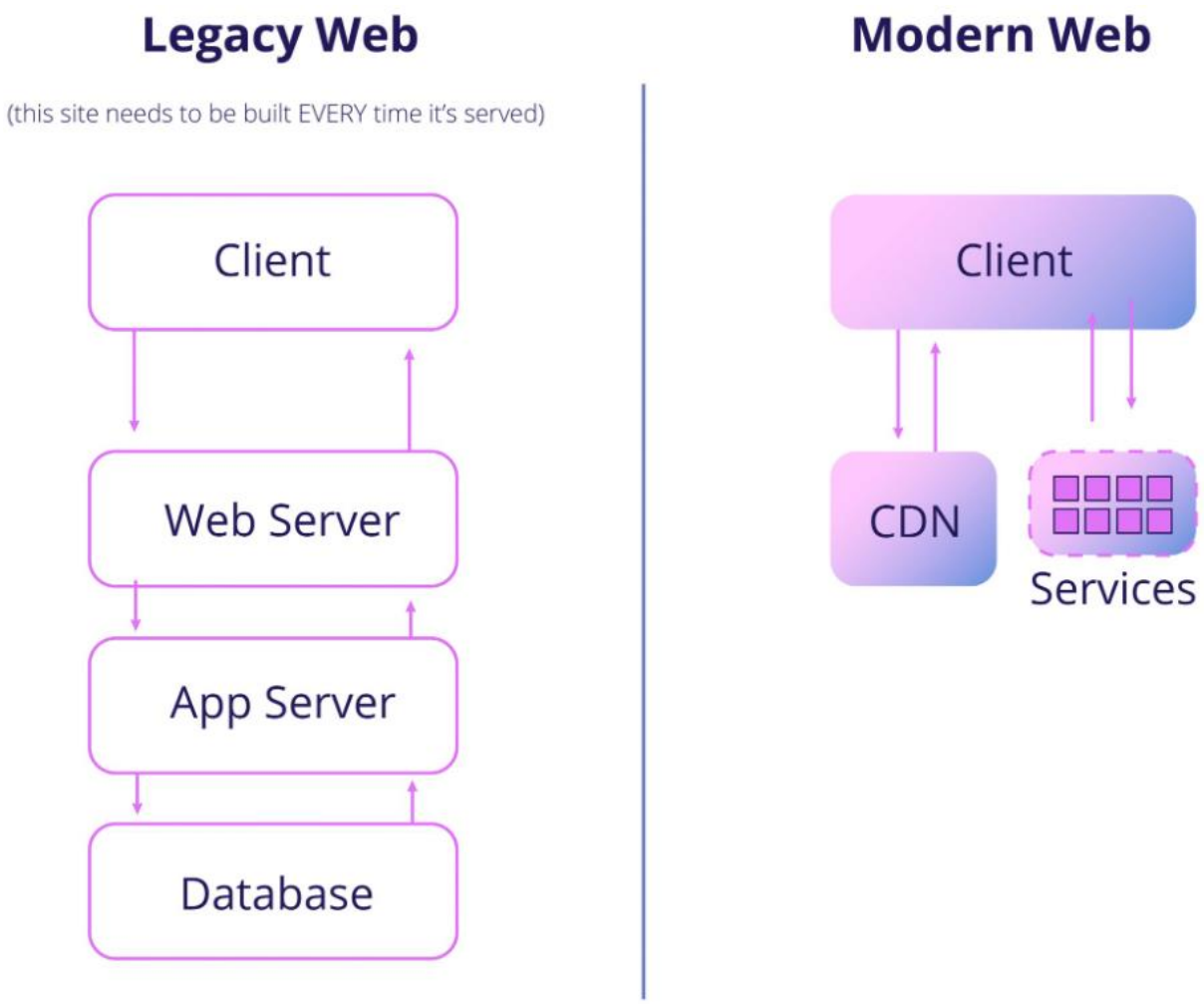
Ha az alkalmazás szerveroldali megjelenítésű, a tartalom a szerverről kerül lekérésre, majd a böngészőnek átadásra, hogy megjelenjen a felhasználónak. Ha a HTML-oldal a kiszolgálóoldalon kerül renderelésre, a felhasználónak az oldalra kell navigálnia, mielőtt a böngésző lehívja az oldalt a kiszolgálóról. Ez azt jelenti, hogy több időre van szükség a tartalom megjelenítéséhez a felhasználó számára. Az oldaltartalom gyorsítótárba helyezéséhez ezt a sémát gyakran az Nginx-szel, egy olyan webkiszolgálóval látják el, amely levélproxyként és terheléselosztóként is használható.

Előnyök és hátrányok

Az a tény, hogy a HTML a szerveren kerül megjelenítésre, számos előnyt biztosít, mint például a SEO, a linkelhetőség és az első oldal azonnali betöltése. A szerver oldali renderelés működik, miközben a JS le van tiltva a böngészőben. Mivel a kódot a szerveren dolgozzák fel, a böngészővel szemben nem támasztanak különleges követelményeket, - ez lehetővé teszi, hogy azonnal észrevegyük a hibákat. Az SSR azonban nem képes kezelni a nehéz szerverkéréseket (ismételt HTML, CSS), ami a szerver betöltésekor vagy egy teljes oldal újraindításakor lassú megjelenítést eredményez. De ennek az alapvető webalkalmazás-architektúra típusnak az igazi Achilles-sarka a gyenge interakció a végfelhasználóval és a teljes értékű felhasználói felület létrehozásának képtelensége. Más szóval, az SSR egyszerű és költséghatékony megoldás, ha egyszerű weboldalt kell készítenie. Ennek az architektúratípusnak a megvalósítása bármilyen programozási nyelvvel és back-enddel lehetséges.

5.2.5.4 Statikus Webhely Generálás (SSG)

A statikus oldalak generálásának folyamata magában foglal egy generátort, amely automatizálja az egyes HTML-oldalak kódolását, és sablonokból hozza létre őket. A Static Site Generation választása esetén egy egyszerű statikus webhely jön létre, amely egy szerveren található, és amely egy már generált HTML-oldalt tartalmaz, amelyet kérésre a felhasználóknak átadnak. Így nem kell minden egyes alkalommal generálni, amikor valaki meglátogatja a Webhelyet - a szerver csak elküldi a már meglévő adatokat egy API-n keresztül.



Előnyök és hátrányok

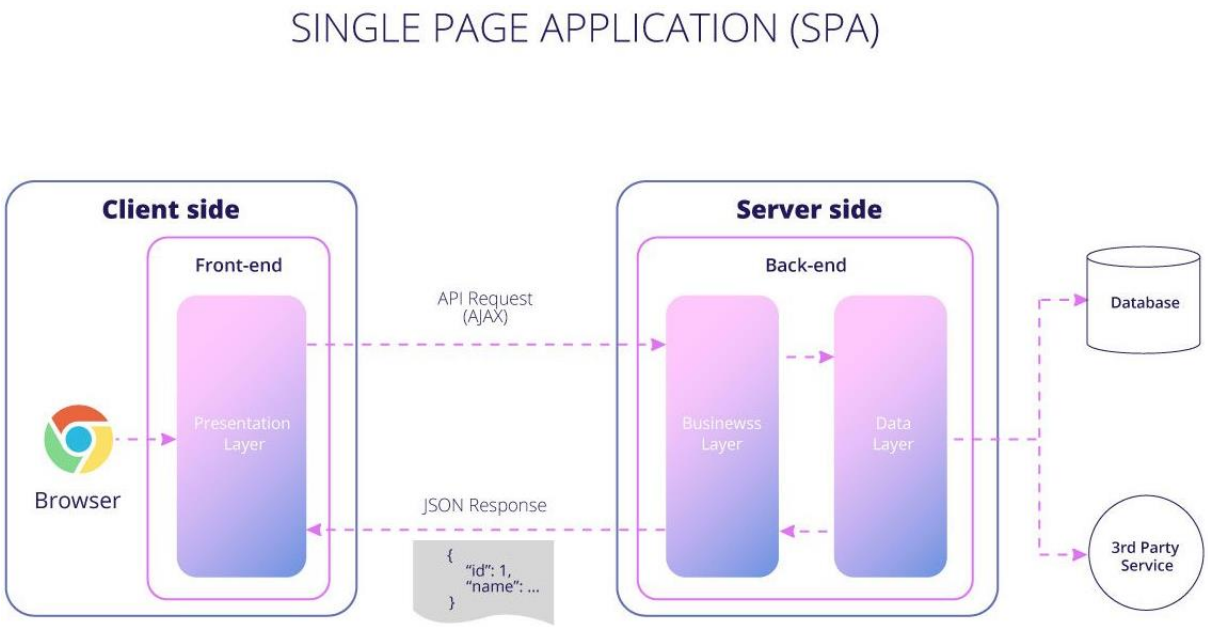
Mindenekelőtt, ez a megközelítés csak weboldalak számára alkalmas. Ezzel együtt a generált webhelyoldalak tartalma nem változik, hacsak nem ad hozzá új adatokat vagy komponenseket. Ez azt jelenti, hogy teljesen újra kell generálnia a webhelyet, amint új tartalmat kíván hozzáadni. Ez az egyik legnagyobb hátránya, amely komolyan korlátozza azokat az üzleti eseteket, amelyekre alkalmazható.

Az előnyök közé tartozik azonban a tartalomelosztó hálózaton (CDN) keresztül szállított statikus tartalom nagy sebessége. Emellett az SSG-ben az összes szerverművelet és az adatbázissal végzett munka egy API-n keresztül valósul meg, amely független a weboldaltól. Ez a lehetőség egyszerűen és így kizárólag megfizethetően megvalósítható.

Egyszerű statikus oldal generátorokra példa a Jekyll és a Hugo, míg az összetettebb megoldások megvalósítására a Gatsby és a VuePress alkalmas .

5.2.5.5 Egyoldalas alkalmazás (SPA)

Az SPA olyan típusú webes alkalmazás, amely egy böngészőben működik. Nem szükséges az oldal újratöltése, amikor új adatokat kell megjeleníteni. Ezt a webalkalmazás-architektúra típust széles körben használjuk a mindennapi életünkben: Facebook, Gmail, Google Maps, GitHub és Twitter - ezek mind egyoldalas alkalmazások.



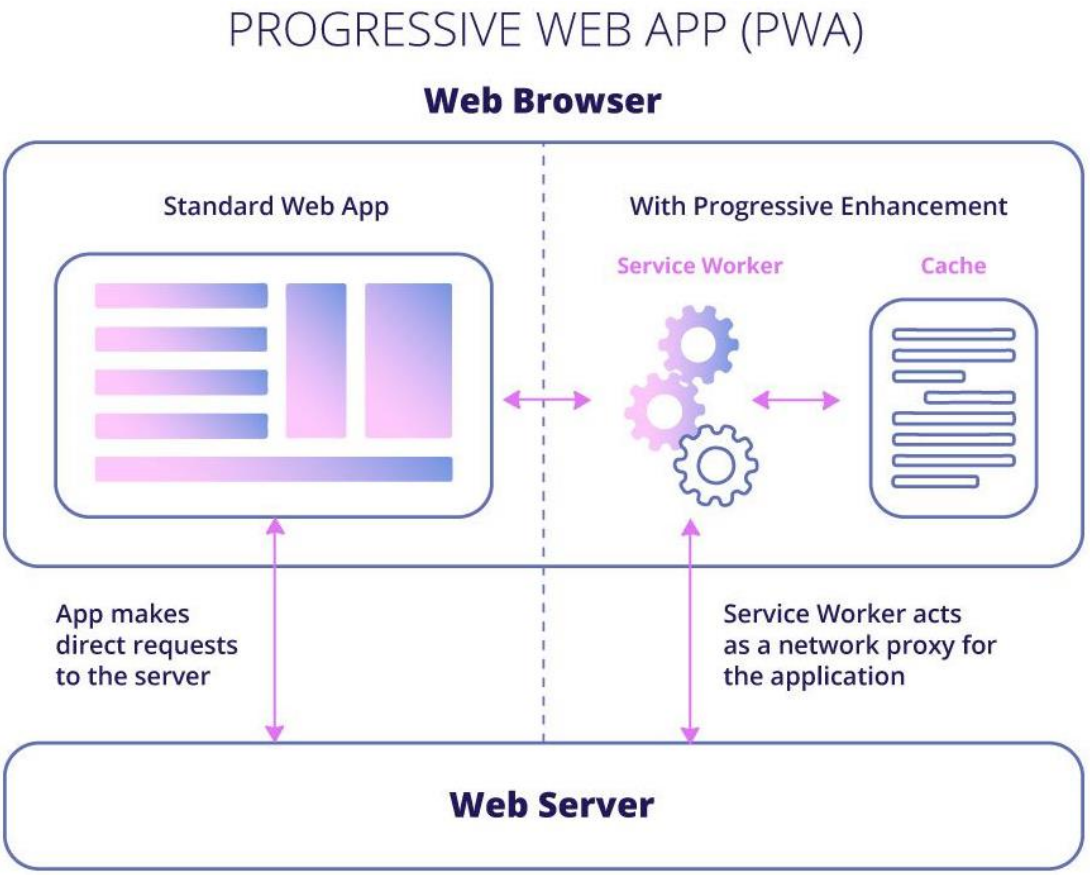
Előnyök és hátrányok

Az SSR-rel és SSG-vel ellentétben az SPA lehetővé teszi, hogy interaktív webes alkalmazást készítsen. API-t használ a szerverrel való kommunikációhoz. Ez az architektúra jó a termék egyszerű skálázásához. Továbbá, ha mobilalkalmazásra van szükség, nincs szükség további erőfeszítésekre az API fejlesztéséhez - a mobilalkalmazás ugyanazt az API-t használhatja, mint a webes.

Az SPA gyors megjelenítést biztosít, miután az alkalmazás teljesen betöltődött a böngészőben, és rendkívül érzékeny szoftvert hoz létre a végfelhasználó számára. Ugyanakkor "megöli" a SEO-t és korlátozza a linkelhetőséget, mivel az ilyen funkciók megvalósítása további erőfeszítéseket igényel. További hátrányai közé tartozik az első betöltéshez szükséges hosszú idő, a rossz útválasztás és az elavult böngészők korlátozott támogatása. Mivel meglehetősen költséges webarchitektúra-típusról van szó, az SPA alkalmas a B2C felhasználóknak szánt reszponzív felhasználói felület létrehozására.

5.2.5.6 Progresszív (PWA)

A progresszív webalkalmazás-architektúra egy egyoldalas webalkalmazás logikáját használja, amely mellett néhány szolgáltatás fut a böngészőben. Ez azt jelenti, hogy a legfontosabb szempont, amit figyelembe kell venni, hogy mind a böngészőnek, mind az operációs rendszernek támogatnia kell ezt a szabványkészletet.



A végfelhasználó számára a progresszív webes alkalmazás fizikailag egy felugró ablakot jelent, amely az alkalmazás hozzáadását kínálja az indítóképernyőn (nem a böngésző, hanem az operációs rendszer képernyőjén), amikor meglátogat egy webhelyet. Ha a felhasználó elfogadja, az alkalmazás automatikusan hozzáadódik az eszközhöz.

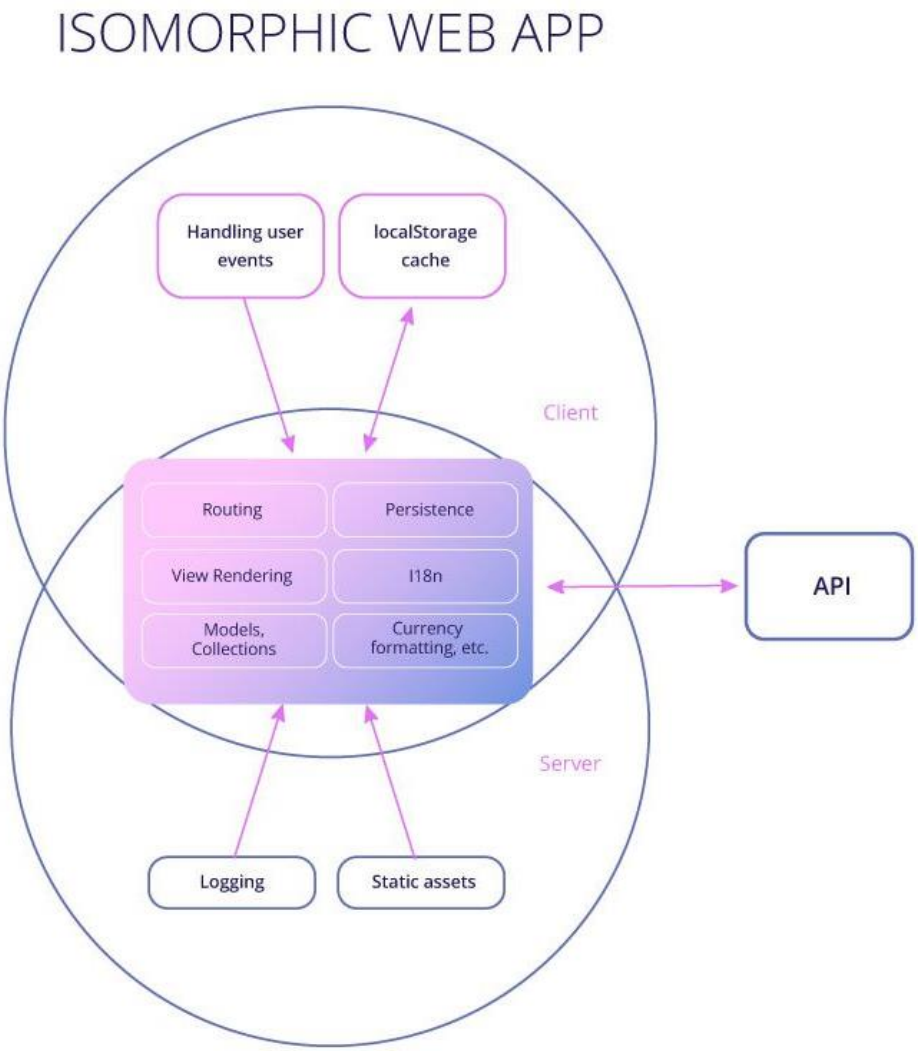
A PWA megvalósítása lehetővé teszi, hogy webes alkalmazása támogassa az offline élményt, a háttérben történő szinkronizálást és a push-értesítéseket. Ez hozzáférést biztosít azokhoz a funkciókhoz, amelyekhez korábban natív alkalmazásra volt szükség. Ugyanakkor a PWA-architektúra kiválasztásakor szem előtt kell tartania, hogy a funkciók többsége nem érhető el iOS-en. Érdemes minden egyes konkrét üzleti esetet elemezni.

Előnyök és hátrányok

A progresszív webalkalmazás architektúrát a Windows, az Android és az iOS támogatja (az iOS esetében az offline mód azonban le van tiltva). A fejlesztők távolról is hozzáadhatnak frissítéseket a webes alkalmazáshoz. A PWA biztonságos, mivel HTTPS-t használ. Ugyanakkor a végfelhasználók anélkül telepíthetnek egy PWA-t, hogy meglátogatnák a Play Marketet vagy az App Store-t. Ennek az architektúratípusnak a hátrányai közé tartozik, hogy olyan böngészőt és operációs rendszert kell választani, amely teljes mértékben támogatja.

5.2.5.7 Izomorf

Egy másik modern webalkalmazás-architektúrát izomorfikusnak neveznek. Ez egy olyan típusú JavaScript-alkalmazás, amely mind a kliens-, mind a szerveroldalon futtatható. Először az ügyfél betölt egy HTML-t, ahol a JavaScript-alkalmazás feltöltődik a böngészőbe, majd az alkalmazás SPA-ként kezd futni.



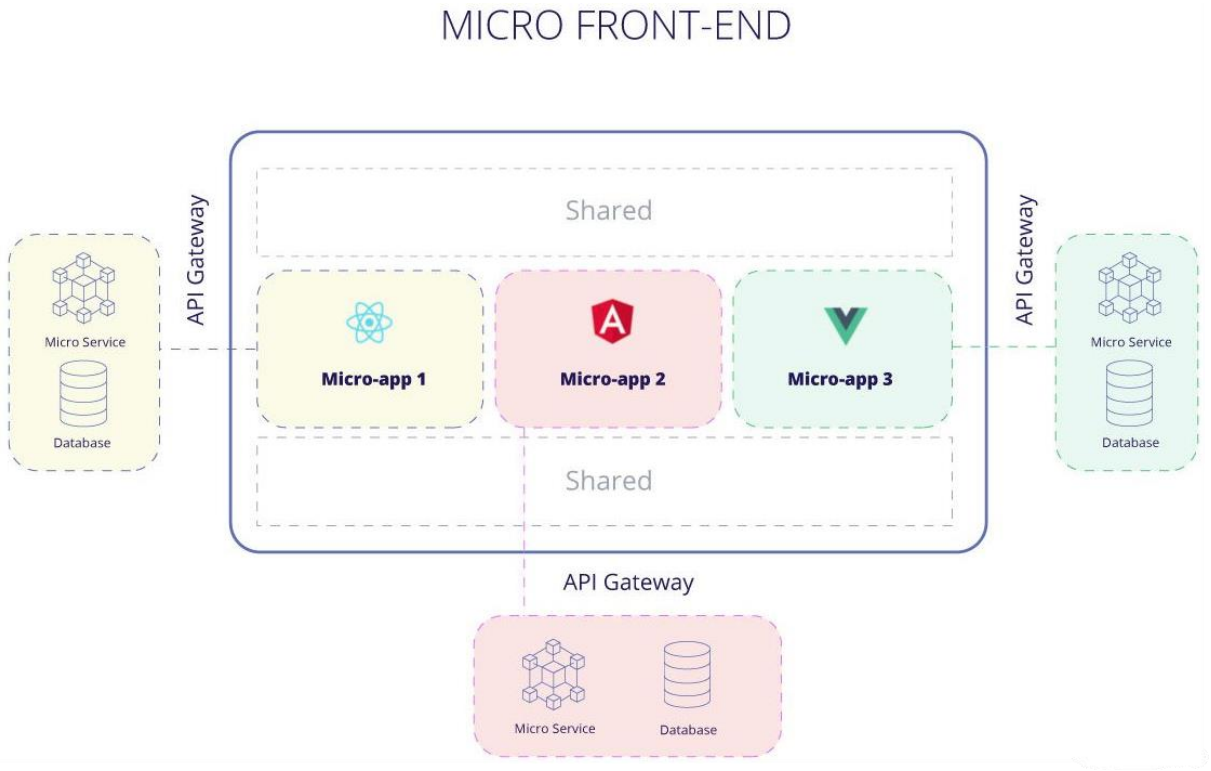
Az SPA-val ellentétben itt az első renderelés a szerveren történik. Más szóval, amikor a felhasználó beírja a webalkalmazás címét, a szerver először teljesen rendereli a HTML-t (JS-kötegeket). Ez lehetővé teszi, hogy a Google lánctalpasok megvizsgálják a weboldalakat, és a webalkalmazás SEO-ja működjön. Ennek eredményeképpen egy vállalkozás egy SPA webalkalmazás már renderelt oldalát és JS kötegeit kapja meg.

Előnyök és hátrányok

A szerveroldali rendereléssel ellentétben az izomorf webes architektúra gyors adatfrissítést, reagálóképességet és több UI/UX lehetőséget biztosít. Gyorsabb renderelést biztosít a szerver betöltésekor, mivel a feldolgozott kód átkerül az ügyfélhez. A kliensoldali rendereléssel ellentétben pedig azonnali megjelenítést biztosít a böngészőben, felhasználóbarát útválasztást, SEO-t és linkelhetőséget. Az egyetlen hátránya ennek a webalkalmazás-architektúra típusnak, hogy teljes mértékben csak a JavaScript támogatja. Ez legtöbbször azt jelenti, hogy a választható tech stack a JS-alapú keretrendszerekre és eszközökre korlátozódik.

5.2.5.8 Micro Front-End

Az egyéb webalkalmazás-tervezési elvek között megkülönböztetjük a micro front-end-et, egy olyan megközelítést, amely a front-end alkalmazás különálló, egymással együttműködő "mikroalkalmazásokra" történő lebontásán alapul. A végfelhasználó számára ezek mindegyike egyetlen oldalon található.



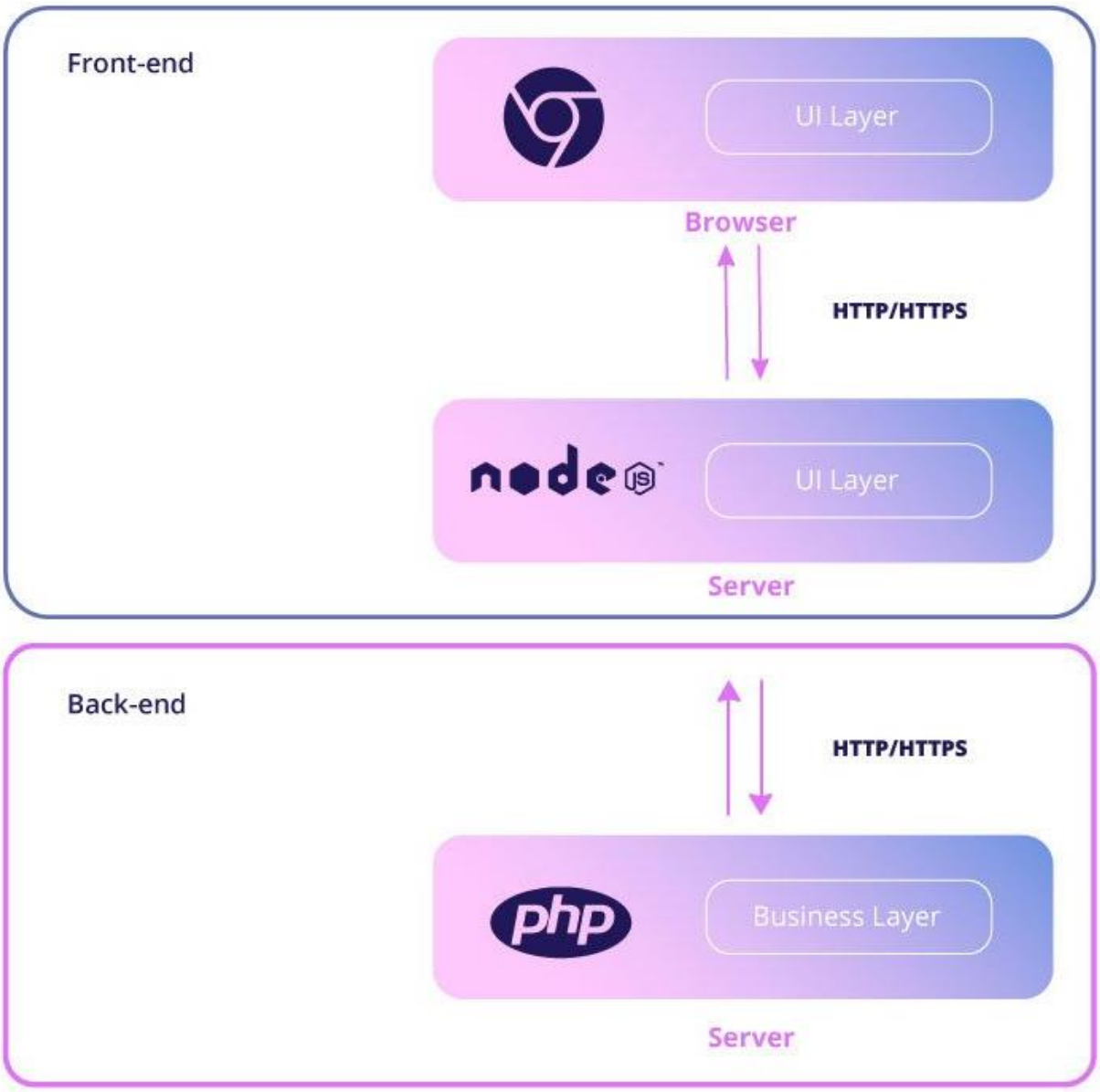
Ez a webalkalmazás-architektúra moduláris, ami azt jelenti, hogy az oldalak és a widgetek teljesen független alkalmazások. Ilyen megközelítéssel a fejlesztés és a telepítés párhuzamosan fut. Ugyanakkor a struktúra bonyolulttá teszi az alkalmazást, és kódduplikációt okoz.

Előnyök és hátrányok

A mikro-frontend architektúra tagadhatatlan előnye a skálázhatóság. A vállalati ügyfelek számára megfelelő, mivel gyakran a hatalmas szoftverrészek fejlesztése több csapat között oszlik meg. A mikroszolgáltatások mind back-, mind frontendben megjelennek, és skálázhatók a csapatok és a felhőben futó terhelés tekintetében.

5.2.5.9 Node.js és az új Webes Front-End

Ez a fajta webalkalmazás-architektúra egy Node.js alapú szerverből és egy felhasználói felületből áll. Ezzel együtt az üzleti logika kiszolgálója bármilyen nyelven írható (vegyük példának a PHP-t), és egy API-t használ a szerverrel való kommunikációhoz.



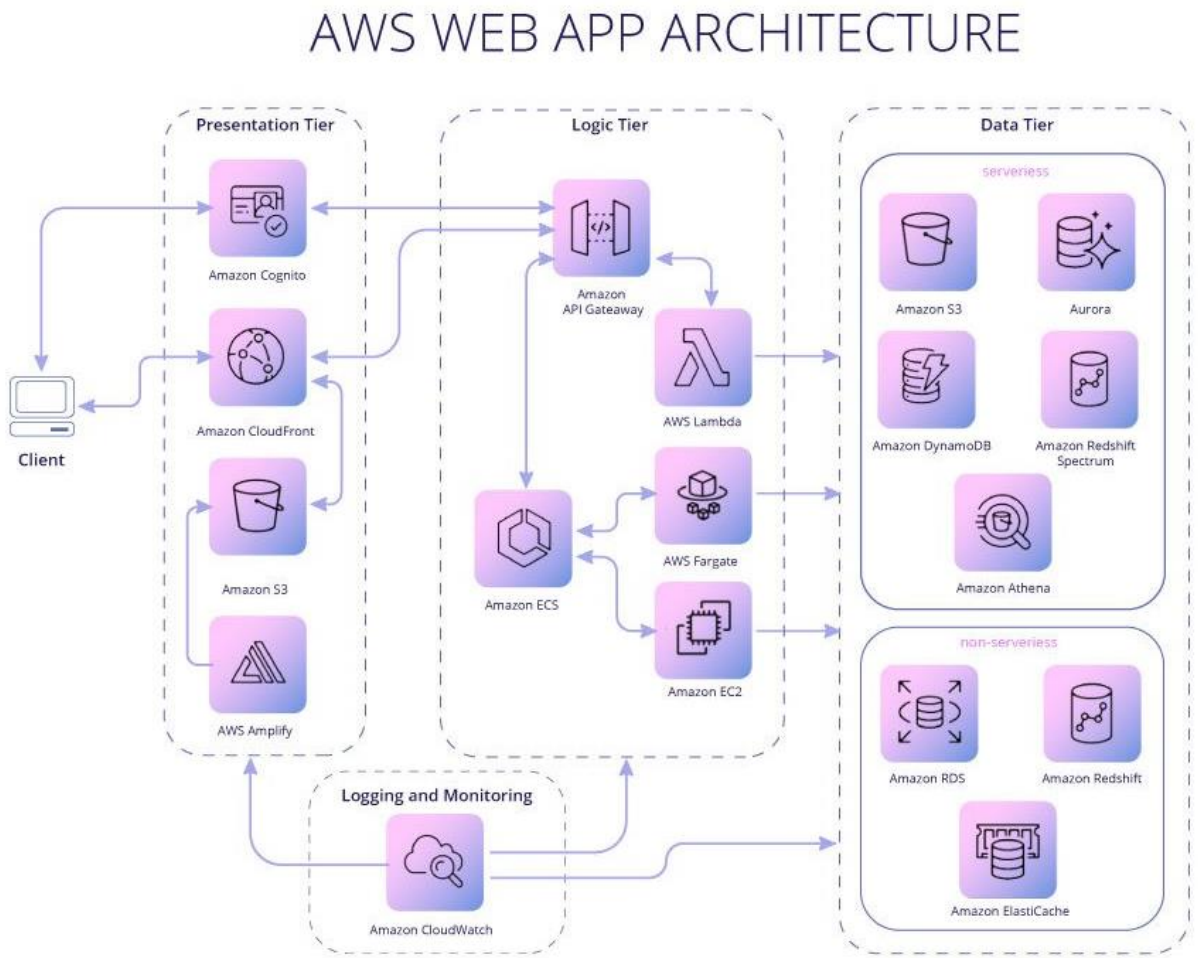
Valójában az új webes front-end megközelítés lehetővé teszi számunkra, hogy kihasználjuk az izomorf architektúra típus, az SSR vagy SPA, a mobil eszközökhöz való API és a linkelhetőség összes előnyét. Az izomorf webes alkalmazásokkal ellentétben nem jelennek meg a nyelvi vagy üzleti logikai platformra vonatkozó korlátozások. Bár az új webes frontend logikát használó webalkalmazás fejlesztése hosszabb időt vesz igénybe, mint az SSG vagy SSR típust használóké. Az új webes frontend koncepciója illik az izomorf webes alkalmazás fejlesztéséhez, ahol már létezik API-kiszolgáló.

5.2.5.10 Webes Felhő (Cloud)

A felhőarchitektúra azt jelenti, hogy a szervereket egy felhőszolgáltató, például az Amazon AWS, Azure vagy Google Cloud kezeli. Amellett, hogy ezek a szolgáltatók a szervereket a saját oldalukon hosztolják, olyan szolgáltatások komplexumát kínálják, amelyek lehetővé teszik a felhőben hosztolt és menedzselt webalkalmazások építését.

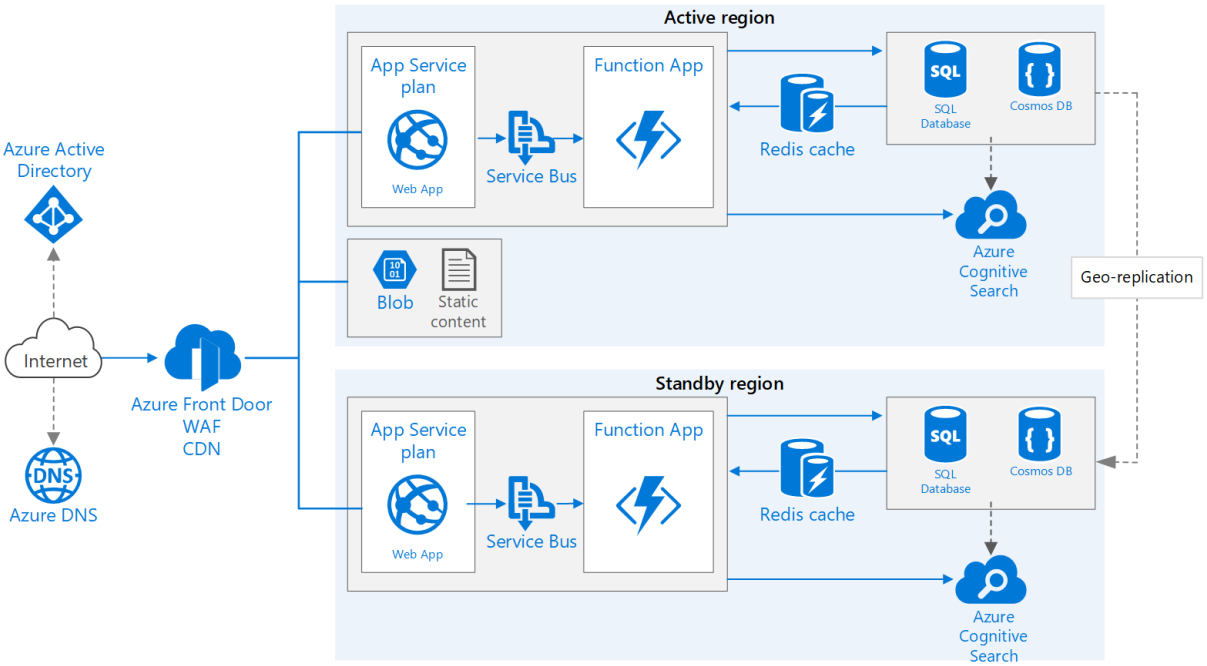
5.2.5.11 AWS (Serverless: szervermentes)

Az AWS szervermentes szolgáltatásai az egyik legnépszerűbb felhőmegoldás, amelyet olyan népszerű minták megvalósítására használnak, mint a mikroszolgáltatások, a mobil háttértárak és az egyoldalas alkalmazások. Az alábbi séma azt mutatja be, hogy az AWS webes szolgáltatásait hogyan lehet használni egy webes alkalmazás létrehozásához a korábban ismertetett 3 szintes architektúra logikáját használva.



5.2.5.12 Azure architektúra

Az Azure számos olyan felhőalapú számítástechnikai szolgáltatást kínál, amelyek lehetővé teszik alapvető és biztonságos kiszolgáló nélküli webalkalmazások létrehozását.



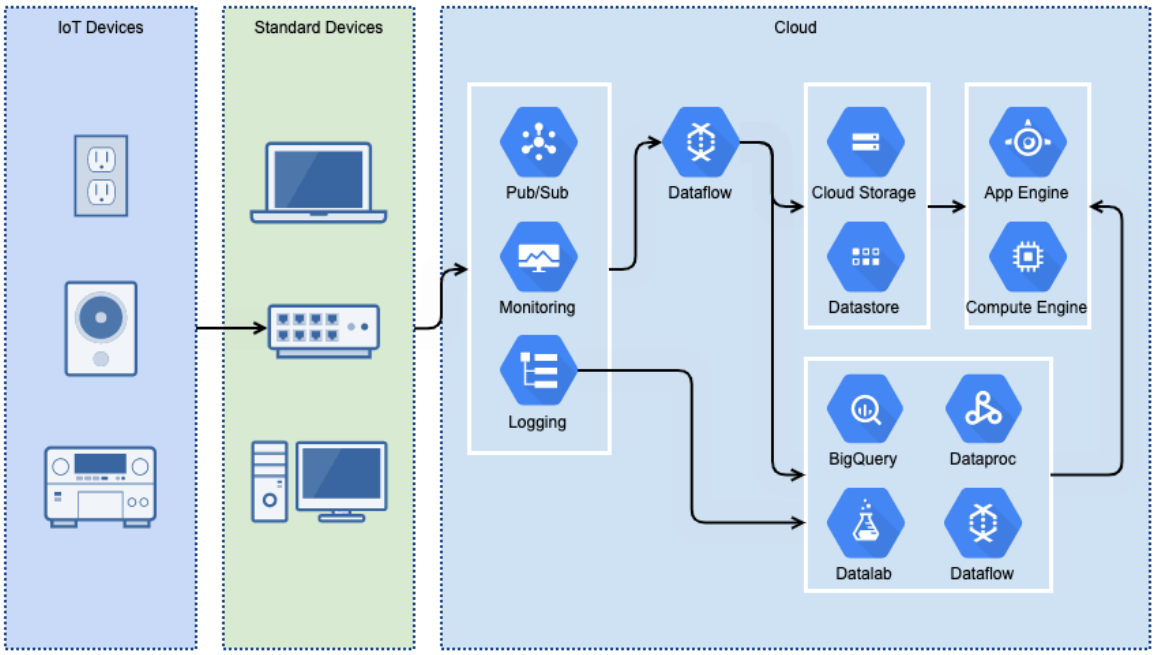
Egy alapvető webes alkalmazás az Azure App Service és az Azure SQL Database segítségével készül. A Web App, Front Door, Function App, Azure DNS, Azure Cognitive Search és Azure DNS kombinációja segít javítani a skálázhatóságot és a teljesítményt egy Azure App Service webes alkalmazásban.

Az Azure segítségével fejlesztett kiszolgáló nélküli webalkalmazás statikus tartalmat szolgál ki az Azure Blob Storage-ból, és Azure Functions segítségével valósít meg egy API-t . Az API adatokat olvas be a Cosmos DB-ből, és az eredményeket visszaadja a webes alkalmazásnak.

Az Azure App Service Environment (ASE) olyan webes alkalmazások telepítésére szolgál, ahol további biztonságra van szükség.

5.2.5.13 Google Cloud Platform

A GCP az üzleti igényektől, valamint a fejlesztői és infrastrukturális csapat érettségétől függően App Engine vagy Cloud Run, Compute Engine vagy Kubernetes Engine szolgáltatást kínál a skálázható architektúrájú webes alkalmazások építéséhez.



5.2.6 Alkalmazás Architektúra – Vállalati szolgáltatás busz

Az ESB (**Enterprise Service Bus**) egy integrációs architektúra, amely a Szolgáltatás orientált Architektúrára (SOA) épül. Az ESB központi szerepet tölt be a különböző rendszerek, alkalmazások és szolgáltatások integrálásában.

Az ESB egy központi helyet biztosít a szolgáltatásoknak, amelyeket az alkalmazások kérhetnek le, vagy amelyeket az alkalmazások kínálnak.

Az ESB architektúra fő rétegei

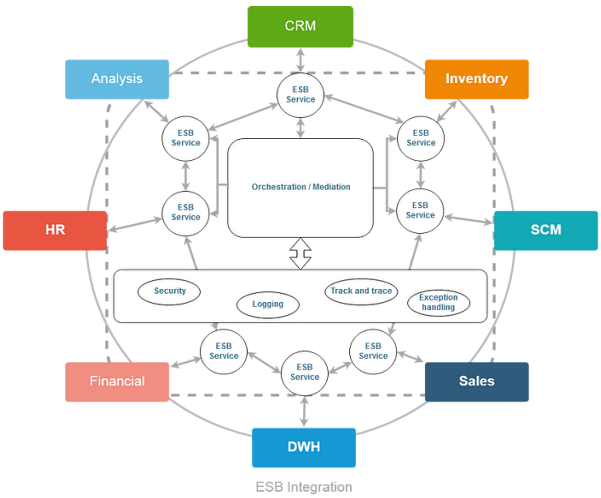
1. **Interfész réteg**: Az interfész réteg az ESB központi része, amely lehetővé teszi az üzenetek továbbítását a különböző rendszerek és alkalmazások között. Az interfész réteg az összes alkalmazás, szolgáltatás és rendszer számára elérhető interfészt biztosít.

2. **Üzleti logika réteg**: Az üzleti logika réteg felelős az adatok feldolgozásáért és az üzleti logika végrehajtásáért. Ez a réteg végzi a szükséges műveleteket az adatok előállításához és feldolgozásához, majd visszaküldi az eredményeket a kérő alkalmazásnak.

3. **Adatelérés réteg**: Az adatelérés réteg az adatok eléréséért és feldolgozásáért felelős. Ez a réteg a különböző adatforrásokhoz kapcsolódik, és lehetővé teszi az adatok megszerzését, tárolását és feldolgozását.

Előnyei közé tartozik a szolgáltatások egyszerűbb kezelése és újabb szolgáltatások könnyebb hozzáadása.

Alkalmazása általában csökkenti az integrációs költségeket, javítja a rendszer rugalmasságát és skálázhatóságát, valamint növeli a rendszer biztonságát és megbízhatóságát.



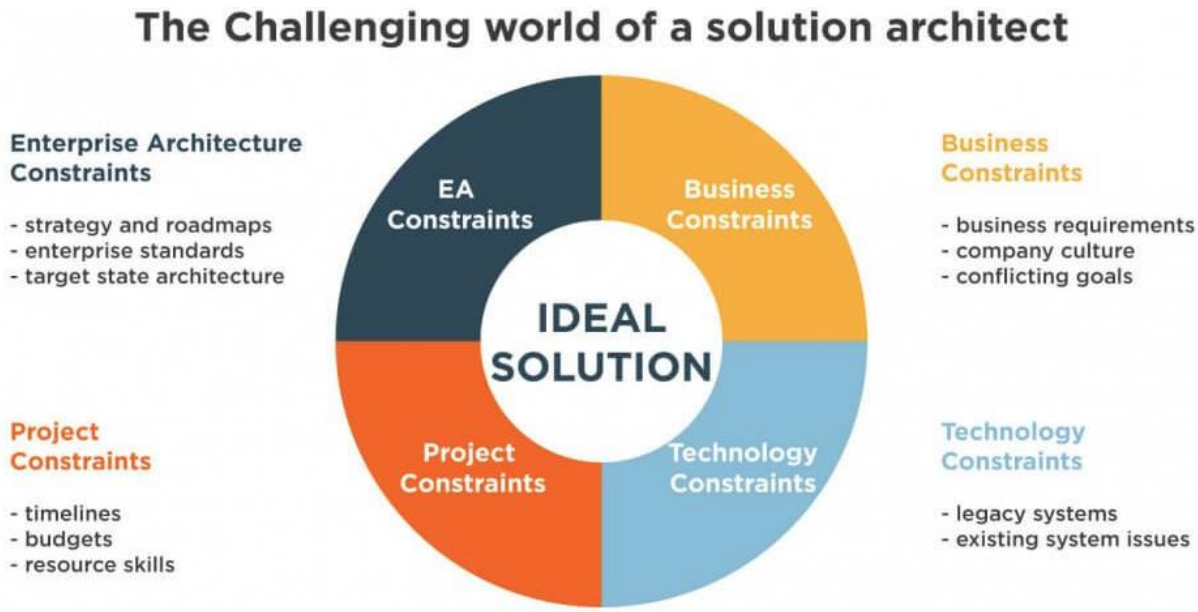
5.3 Megoldás Architektúra – Solution Architecture

A Megoldás Architektúra tervezés célja, hogy meghatározza az informatikai rendszerek és alkalmazások hatékony működésének kialakításához szükséges feltételeket, az adott problémára a legjobb megoldást, figyelembe véve a vállalat céljait, üzleti igényeit, technológiai lehetőségeit és korlátait.

A tervezés során a tervezőknek figyelembe kell venniük a rendszer architektúrájának összes elemét, beleértve az infrastruktúrát, az adatokat, az alkalmazásokat és a felhasználói felületet.

A megoldás architektúra tervezése általában a projekt tervezés egyik legkorábbi és legfontosabb lépése, amelynek során a tervezők átfogó képet kapnak a problémáról, és megtervezik a megoldást, mielőtt az alkalmazás vagy rendszer fejlesztése elkezdődik.

A Megoldás Tervező (**Solution Architect**) szerepkör az Alkalmazás architektúra és az Üzleti architektúra közötti szint, amely az informatikai rendszerek, alkalmazások és szolgáltatások egészének tervezésével foglalkozik.



5.3.1 Megoldás Architektúra Szint

Megoldástervezés

Az üzleti problémákra adott megoldásokat határozza meg, és az alkalmazások, szolgáltatások és rendszerek összekapcsolását tervezi meg.

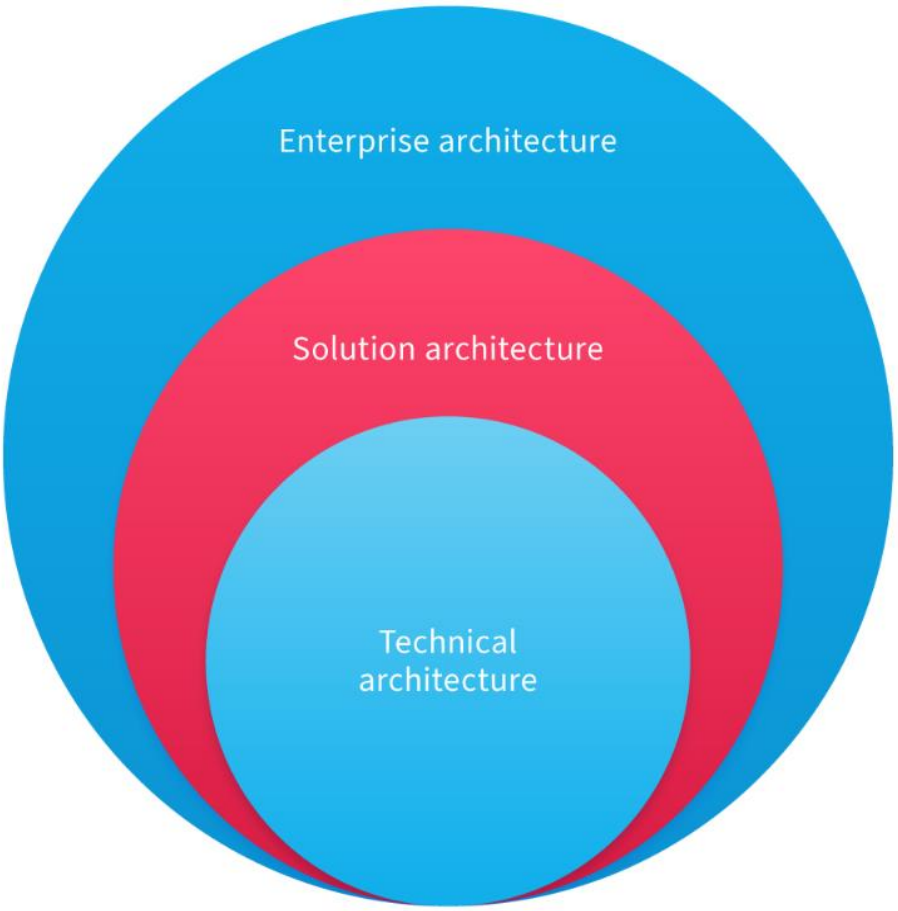
Implementációs tervezés

Az informatikai rendszer tervezését, az alkalmazások és szolgáltatások implementálását és integrációját foglalja magában.

Teszttervezés

A tesztelési folyamatok tervezését jelenti, beleértve az egyes alkalmazások, szolgáltatások és rendszerek tesztelését.

A Megoldás Tervező (Solution Architect) kompetenciái



5.3.2 . Megoldás Architektúra Minta (Pattern)

Solution Architecture minták olyan bevált gyakorlatokat, tervezési mintákat és elveket jelentenek, amelyek használata segíti a szoftverfejlesztőket és az architektusokat az üzleti igényekre adott megfelelő válaszok megtalálásában. Ezek a minták általában széles körben elfogadott és elismert tervezési sablonokat, mintákat és megoldásokat foglalnak magukba, amelyek segítik az architektusokat a szoftverrendszerek tervezése és implementálása során.

A Solution Architecture minták nagyon sokféle területre vonatkozhatnak, a hardver- és szoftverrendszerektől az adatbázis-kezelő és integrációs rendszereken át a felhőalapú infrastruktúrákig.

Solution Architecture tervezési minták

- **Mikroszolgáltatás-architektúra minták**
- **RESTful API minták**
- **Cloud-native alkalmazások tervezési mintái**
- **Adatintegrációs minták**
- **Biztonsági minták**
- **Skálázhatósági minták**
- **Teljesítmény-optimalizációs minták**
- **Rendszerintegrációs minták**

A Solution Architecture minták célja, hogy segítsék az architektusokat és a szoftverfejlesztőket az optimális és hatékony megoldások megtalálásában, és javítsák az alkalmazások minőségét és teljesítményét.

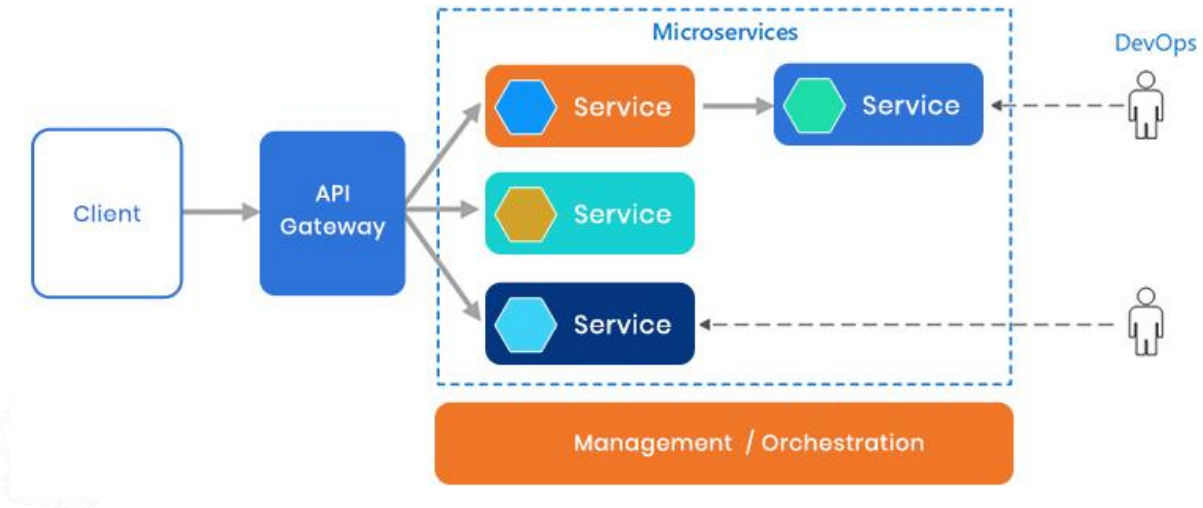
5.3.3 . Megoldás Architektúra Minta - Mikroszolgáltatás

A Mikroszolgáltatás Architektúra tervezési minta egy olyan megközelítés, amely a szoftvertervezés során az alkalmazást több, kisebb és függetlenül fejleszthető komponensre bontja. Az ilyen mikroszolgáltatások szorosan összekapcsolódnak egymással, és összeköttetéseiket az ESB (**Enterprise Service Bus**) vagy más üzenetküldő rendszerek segítik.

Ez a megközelítés a hagyományos monolitikus architektúrával szemben a rugalmasságot, a könnyű karbantarthatóságot és a skálázhatóságot állítja előtérbe. Minden mikroszolgáltatás külön-külön is telepíthető és üzemeltethető, így könnyű az életciklusuk kezelése. Az egyes mikroszolgáltatások lehetnek különböző technológiákban megvalósítottak, így a fejlesztők a legjobb megoldásokat választhatják.

Az ilyen típusú architektúra előnyei közé tartozik a fejlesztés sebessége, az üzleti folyamatok egyszerűbb karbantartása, valamint az alkalmazás nagyobb skálázhatósága és megbízhatósága.

Azonban a megfelelő mikroszolgáltatás-architektúra kialakítása összetett lehet, és a megoldás architekti feladata, hogy figyelembe vegye az üzleti igényeket és az alkalmazás környezetét a megfelelő tervezési döntések meghozatalához. A megoldás architektusnak emellett az is fontos feladata, hogy az egyes mikroszolgáltatásokat összekapcsolja, és biztosítsa az egységes kommunikációt és adatkezelést a rendszer egészében.



5.3.4 Megoldás Architektúra Minta - RESTful API

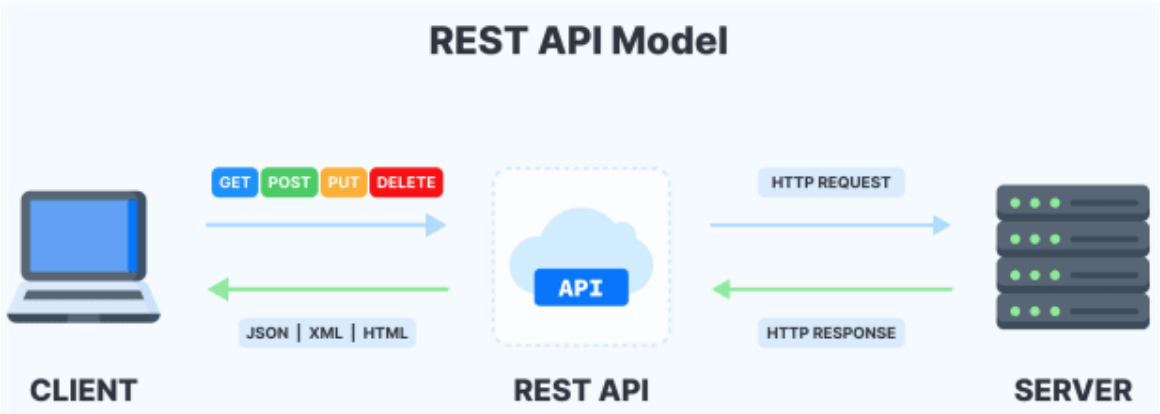
A REST (**Representational State Transfer**) egy Webes szolgáltatások tervezésére és implementálására használt architekturális stílus, amely két fő komponensből áll: a kliens és a szerver közötti kommunikációra szolgáló **protokollból** és a kliens által használt **erőforrásokból**. A RESTful API (**Application Programming Interface**) olyan API, amely a REST architekturális stílusán alapul.

A RESTful API lehetővé teszi az alkalmazások és szolgáltatások közötti kommunikációt, és az adatok átvitelére a HTTP protokollt használja. Az API-n keresztül az alkalmazások elérhetik a szolgáltatásokat, és lehetővé teszi a különböző platformok közötti integrációt.

A RESTful API-nak jellemzői

- **Állapotmentesség**: A kérésnek és a válasznak tartalmaznia kell minden információt, amelyre szükség van a kommunikációhoz, így nem szükséges tárolni semmilyen információt a szerver oldalon.
- **Erőforrás-orientáltság**: A szolgáltatások az erőforrásokra vonatkozó műveletekkel kommunikálnak, amelyek az URL-ekben vannak meghatározva.
- **Egyértelmű interfész**: Az API-nak egyértelmű interfészt kell biztosítania az alkalmazások számára, így azok könnyen és egyszerűen használhatják a szolgáltatásokat.
- **Hierarchikus rendszer**: Az erőforrásokat hierarchikus rendszerben kell tárolni és megjeleníteni.
- **Önleíró üzenetek**: Az üzeneteknek tartalmazniuk kell minden információt, amelyre szükség van az üzenet feldolgozásához.

A RESTful API-kat sokféle szolgáltatás használja, például az **Amazon Web Services**, a **Google Maps**, a **Twitter** és a **Facebook** is.



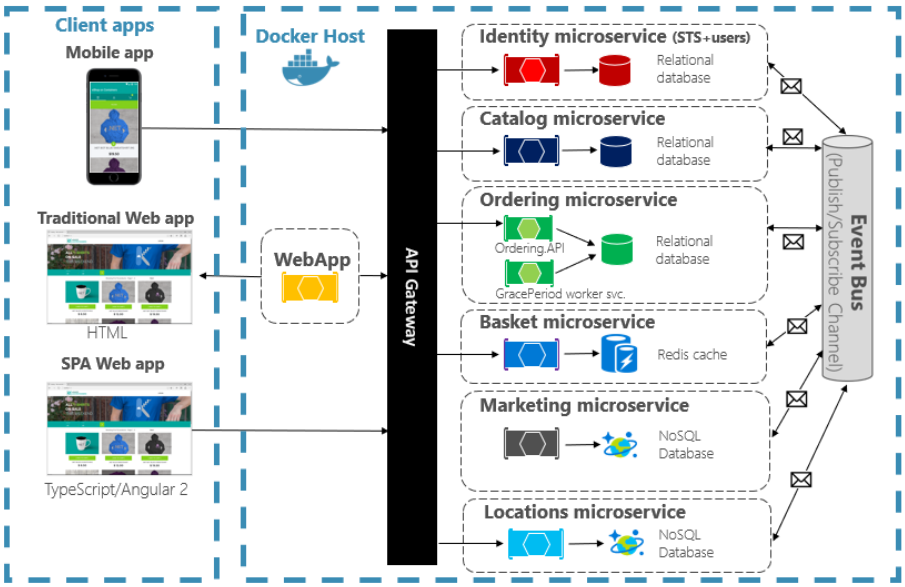
5.3.5 Megoldás Architektúra Minta - Cloud-native alkalmazás

A Cloud-native megoldás architektúra tervezési minta a modern alkalmazások fejlesztése és üzemeltetése során alkalmazott megközelítés, amelynek célja a felhőalapú infrastruktúrákra történő hatékony és skálázható alkalmazásfejlesztés. A Cloud-native architektúra kialakítása során az alkalmazások függetlenül működnek a felhőalapú infrastruktúrától, és maximálisan kihasználják az adott platform lehetőségeit.

A Cloud-native megoldások jellemzői

- **Modularitás**: az alkalmazásokat kisebb, önálló modulokra bontják fel, amelyek könnyen karbantarthatók és skálázhatók.
- **Rugalmasság**: az alkalmazások alkalmazkodnak a változó igényekhez, skálázhatók, és gyorsan reagálnak az eseményekre.
- **Függetlenség a platformtól**: az alkalmazások platformfüggetlenek, és maximálisan kihasználják a felhőalapú infrastruktúra előnyeit.
- **Automatizálás**: a Cloud-native alkalmazások automatizáltak, és kevés emberi beavatkozást igényelnek a fejlesztés, tesztelés és üzemeltetés során.
- **Mikroszolgáltatások**: az alkalmazásokat különböző, önállóan skálázható mikroszolgáltatásokra bontják fel.

A Cloud-native megoldások alkalmazása jelentős előnyökkel jár a hagyományos fejlesztési módszerekhez képest. Az alkalmazásokat rugalmasabban lehet skálázni és karbantartani, csökkentve ezzel az infrastruktúrára fordított költségeket. Emellett a Cloud-native alkalmazások jobban alkalmazkodnak a változó üzleti igényekhez, és gyorsabban reagálnak az eseményekre.



5.3.6 Megoldás Architektúra Minta – Cloud konténerizáció

A mikroszolgáltatások és a konténerek alapú architektúra az elosztott, kritikus fontosságú alkalmazások építésének új megközelítése. Az elképzelés lényege, hogy az alkalmazásokat nem egyetlen és monolitikus egységként, hanem független, egymással együttműködő szolgáltatások gyűjteményeként kell újra gondolni.

A konténerizáció vagy konténeralapú virtualizáció egy operációs rendszer szintű virtualizációs módszer az elosztott alkalmazások telepítésére és futtatására anélkül, hogy minden egyes alkalmazáshoz virtuális gépeket kellene indítani.

Minden mikroszolgáltatás megírható és megvalósítható a legmegfelelőbb nyelvvel vagy technológiával, kihasználva a konténerizáció képességét. Ezáltal a lehető legjobban megfelel az adott igényhez kapcsolódó sajátosságoknak és követelményeknek. Ezzel az új architekturális paradigmával minden szükséges rugalmassággal rendelkezhetünk egy vállalati alkalmazás fejlesztéséhez anélkül, hogy a monolitikus alkalmazásokban általában jelenlévő kompromisszumokat kellene kötnünk.

A Konténer mint szolgáltatás megjelenése

A PaaS és az IaaS architektúrájában a PaaS az IaaS fölött helyezkedik el. A konténerek bevezetésével egy **új köztes réteg alakult ki, a „ Containers as a Service”,** ami lehetővé teszi:

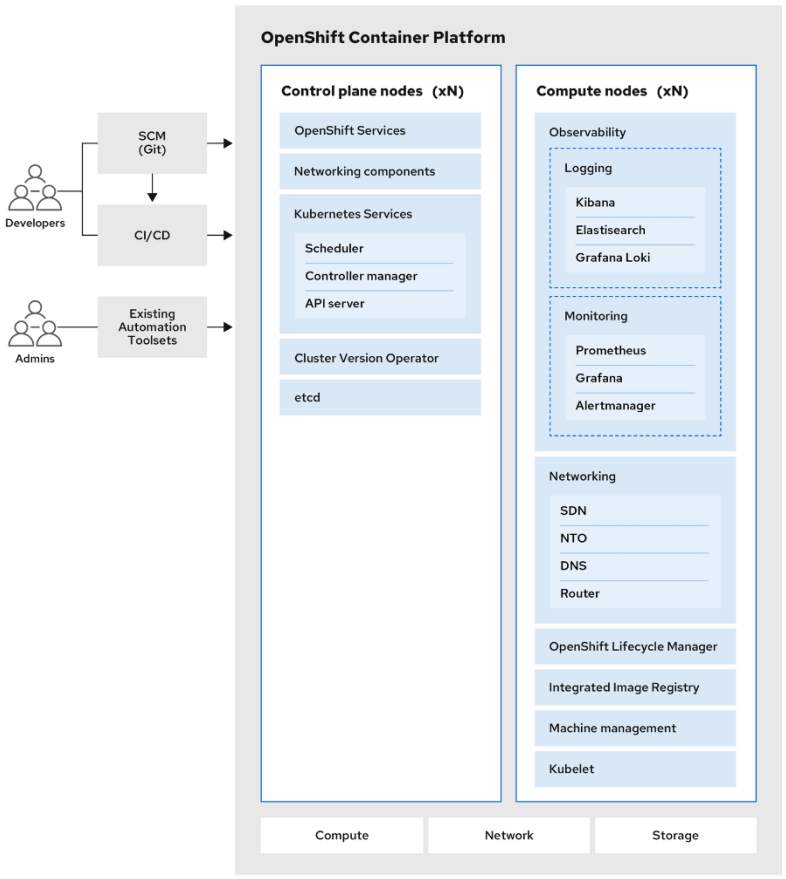
- hogy egy alkalmazást és függőségeit erőforrás-elszigetelt folyamatok futtassák
- a jobb alkalmazásfejlesztést
- csökkenthetik a működési költségeket
- a konténerekben működő alkalmazások gyorsan telepíthetők különböző operációs rendszerekre.

A Konténer-platform megjelenése

A konténerizációt támogató legismertebb szoftverek a Docker, a Kubernetes, a Red Hat OpenShift Container Platform a konténeres alkalmazások fejlesztésére és futtatására alkalmas. Úgy tervezték, hogy az alkalmazások és az azokat támogató adatközpontok néhány gépről és alkalmazásról több ezer gépre bővülhessenek, amelyek több millió ügyfelet szolgálnak ki.

A Konténer Platform ugyanazt a technológiát tartalmazza, amely a hatalmas távközlési, streaming video-, játék-, banki és egyéb alkalmazások motorjaként szolgál. A nyílt Red Hat technológiákban való megvalósítása lehetővé teszi, hogy a konténerizált alkalmazásokat kiterjessze az egyetlen felhőn túl a helyben lévő és a több felhőből álló környezetekre is.

Bár a konténerképek és a belőlük futtatott konténerek a modern alkalmazásfejlesztés elsődleges építőkövei, ezek méretarányos futtatásához megbízható és rugalmas elosztórendszerre van szükség. A Kubernetes a konténerek „orchestrálásának” de facto, nyílt forráskódú szabványa, ami mindössze néhány év alatt hatalmas felhő- és helybeni elfogadottságot ért el.



OpenShift konténer Platform

A konténeres alkalmazások előnyei

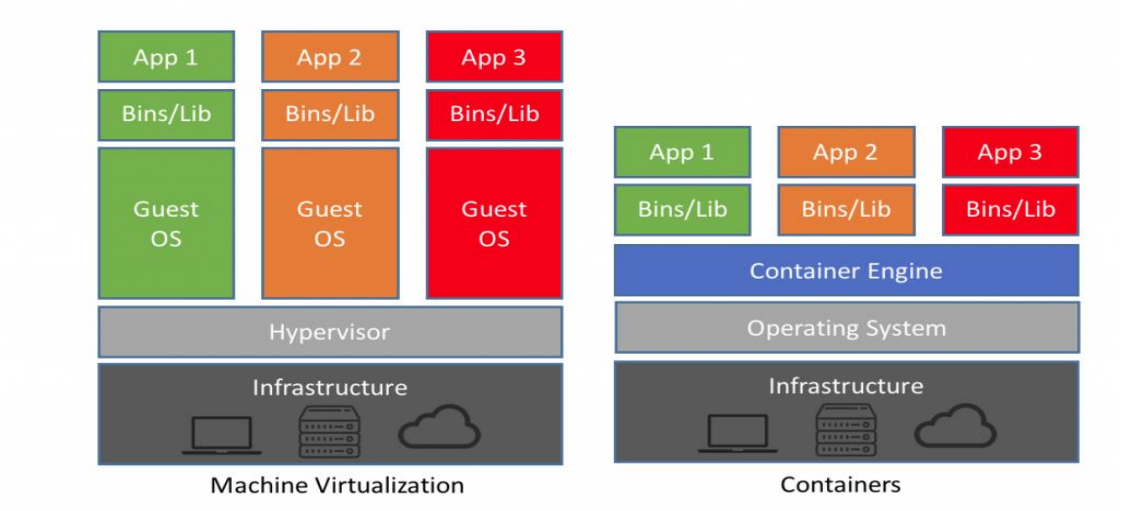
A konténerizált alkalmazások tervezése és használata számos előnnyel jár a hagyományos telepítési módszerek alkalmazásával szemben. Míg korábban elvárták, hogy az alkalmazásokat az összes függőségüket tartalmazó operációs rendszerre telepítsék, a konténerek lehetővé teszik, hogy az alkalmazás magával vigye a függőségeit. A konténerizált alkalmazások létrehozása számos előnnyel jár.

A konténerek kis, dedikált Linux operációs rendszereket használnak kernel nélkül. A fájlrendszerük, a hálózat, a cgroups, a processztáblák és a névterek elkülönülnek a gazdanevű Linux rendszertől, de a konténerek szükség esetén zökkenőmentesen integrálódnak a gazdanevű rendszerbe. A Linux-alapúság lehetővé teszi a konténerek számára, hogy kihasználják a nyílt forráskódú fejlesztési modell gyors innovációval járó összes előnyét. Mivel minden konténer dedikált operációs rendszert használ, az egymással ütköző szoftverfüggőségeket igénylő alkalmazások ugyanazon a gazdán telepíthetők. Minden konténer a saját függő szoftverét hordozza, és saját interfészeit kezeli, például a hálózati és fájlrendszereket, így az alkalmazásoknak soha nem kell versenyezniük ezekért az eszközökért.

Mivel egy alkalmazás összes szoftverfüggősége magában a konténerben oldódik meg, az adatközpont minden egyes állomásán szabványosított operációs rendszert használhat. Nem kell külön operációs rendszert konfigurálnia minden egyes alkalmazásgazda számára. Ha az adatközpontjának nagyobb kapacitásra van szüksége, telepíthet egy másik általános állomásrendszert.

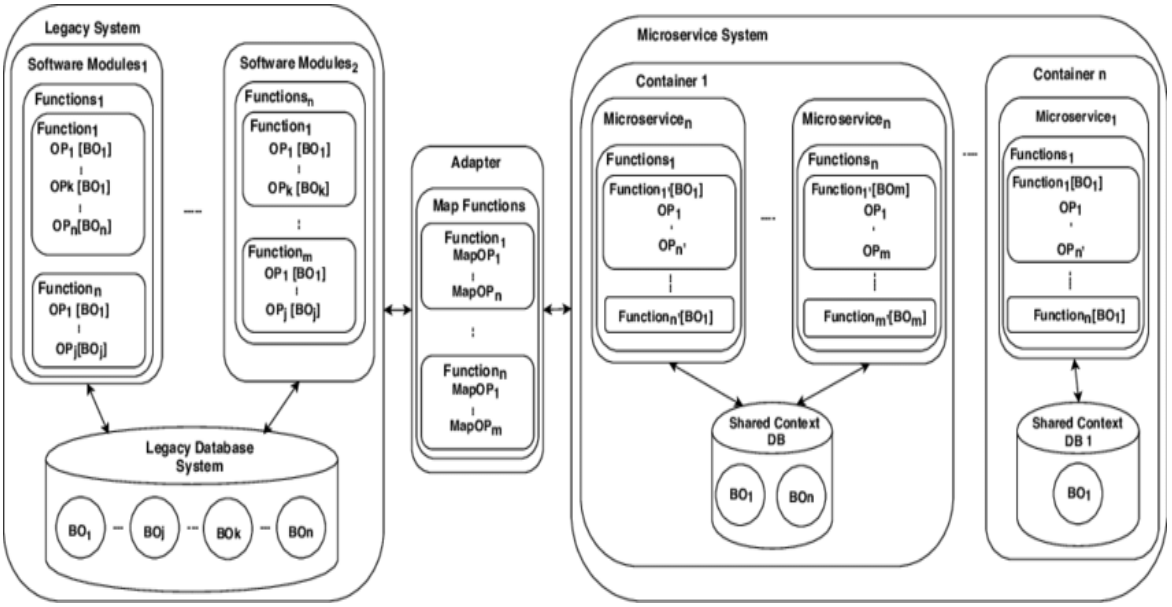
A Konténer Platform különböző vállalati szintű fejlesztéseket is kínál a Hibrid felhőkhöz fürtöket lehet telepíteni a különböző nyilvános felhőplatformokra vagy az adatközpontokba.

A virtuális gép vs Konténeres technológia alapjai



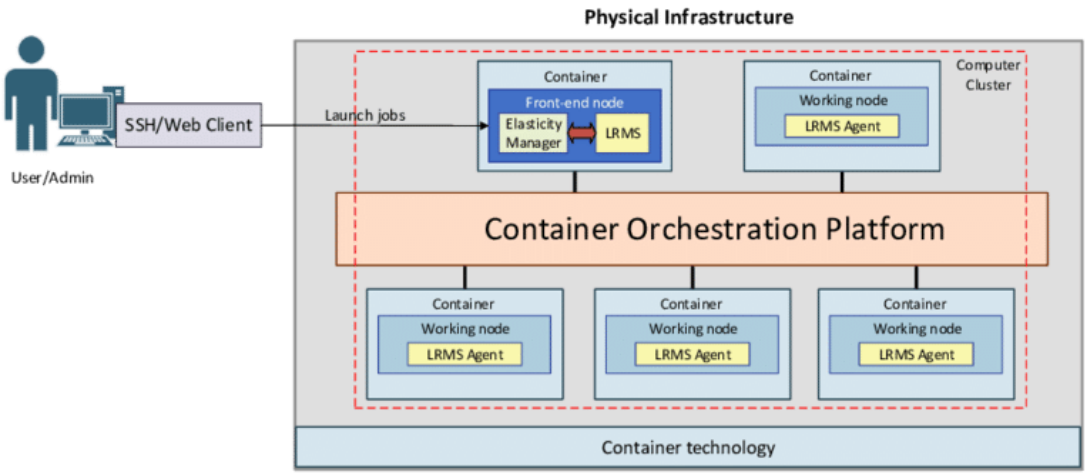
Virtuális gép vs. konténeres technológia architektúra felépítése

A konténeres rendszer architektúra tervezése

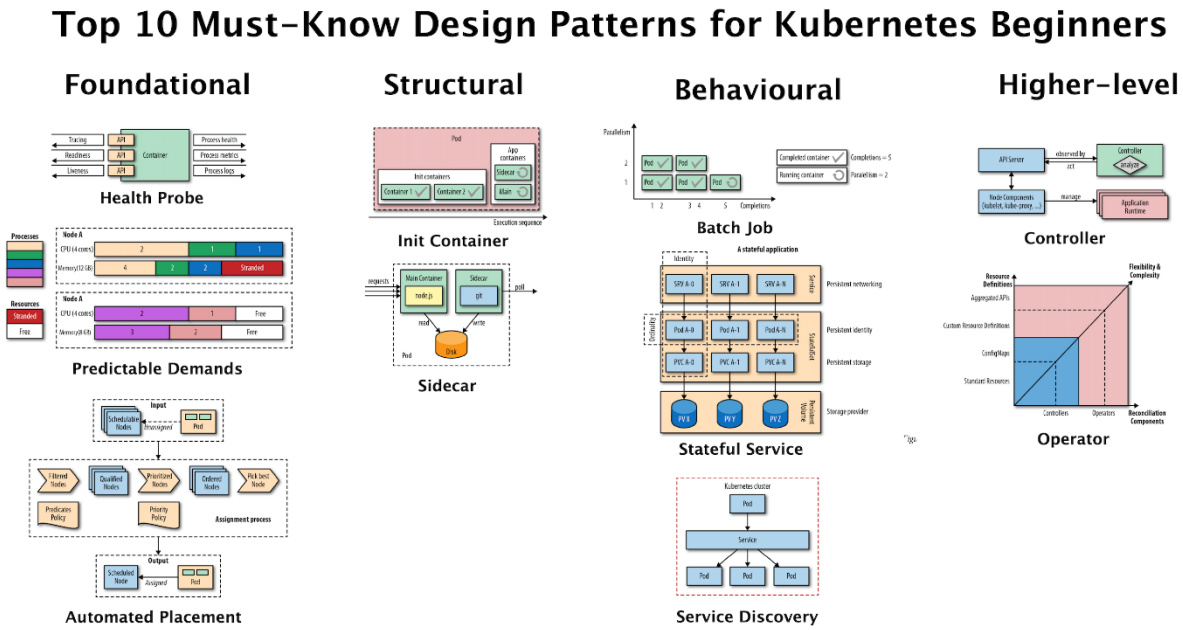


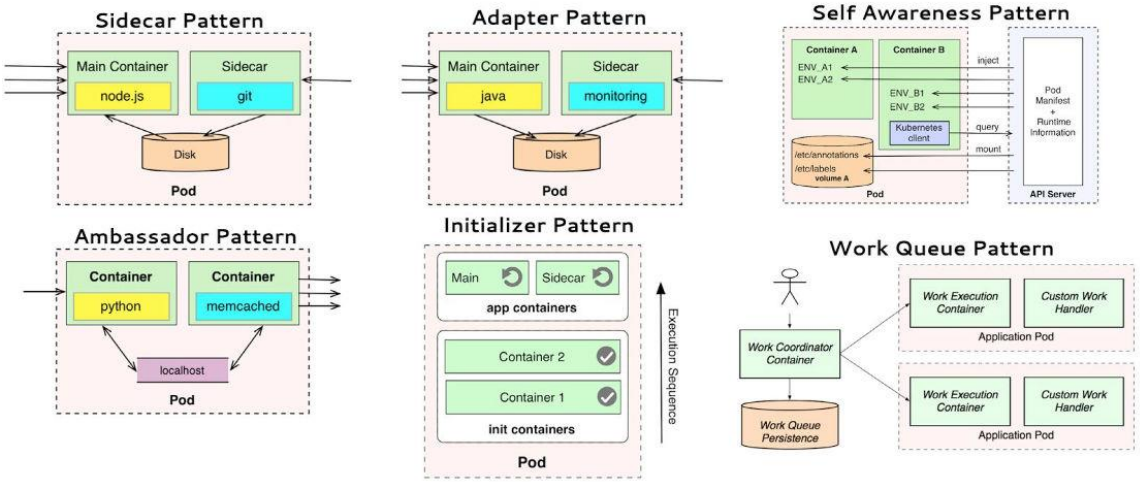
Hagyományos vállalati és mikroszolgáltatási rendszerek architektúrája

A konténeres rendszerek tervezése során kialakított dekomponált funkcionalitás mentén meghatározott mikroszerviz-konténerek közötti folyamatokat már bonyolult orkesztrációs platform vezérli:



A tervezéshez már számos konténerezett mikroszerviz design pattern, tervezési minta áll a tervezők rendelkezésére.





5.4 Szoftver Architektúra

A szoftverarchitektúra a szoftverrendszerek nagyobb szintű tervezése és szervezése. Az architektúra magába foglalja a rendszer fontos elemeinek azonosítását, valamint ezek összekapcsolását és koordinálását úgy, hogy az egész rendszer összhangban működjön. Az architektúra határozza meg a rendszer elemeinek a felépítését, azok kommunikációját, illetve az adatok és folyamatok áramlását a rendszerben. A jó szoftverarchitektúra lehetővé teszi a rendszer könnyű karbantartását, bővítését és skálázhatóságát.

Rétegek

A szoftver architektúra általában különböző rétegekre van tagolva, amelyek segítenek az alkalmazás logikájának és fejlesztési folyamatának szervezésében és megértésében.

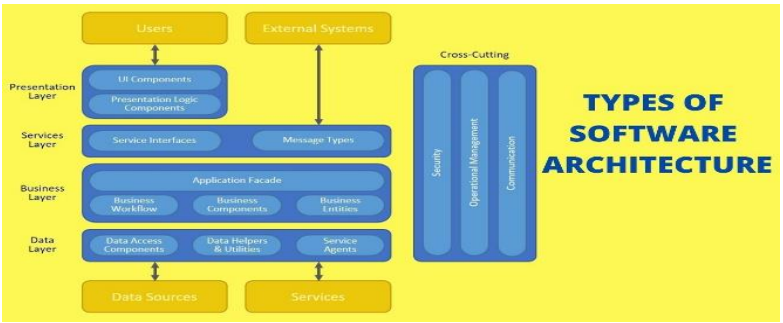
1. **Prezentációs réteg**: Ez a réteg az alkalmazás felhasználói felületét jelenti, amely lehet webes vagy desktop alkalmazás. Ez a réteg határozza meg, hogyan jelennek meg az adatok a felhasználók számára és hogyan tudnak interakcióba lépni velük.

2. **Szolgáltatás réteg**: Ez a réteg a rendszer használói számára elérhető szolgáltatások, rendszerhozzáférési pontok, funkciók felületét biztosítja.

3. **Üzleti logikai réteg**: Ez a réteg tartalmazza az üzleti logikát, amely meghatározza, hogy az adatok hogyan vannak feldolgozva és milyen műveletekkel végzik el azokat. Ez a réteg meghatározza az alkalmazás viselkedését és funkcióit.

4. **Adatkezelési réteg**: Ez a réteg az adatokhoz kapcsolódó műveleteket tartalmazza, például az adatok lekérdezése, beillesztése, frissítése és törlése. Ez a réteg lehetővé teszi az adatok megbízható és hatékony kezelését.

5. **Infrastruktúra réteg**: Ez a réteg tartalmazza az olyan alapvető szolgáltatásokat, mint az adatbázis-kezelő rendszerek, az operációs rendszerek, a hálózati szolgáltatások stb. Ez a réteg a többi réteg megfelelő működéséhez szükséges.



5.4.1 Környezet

- Referencia modell

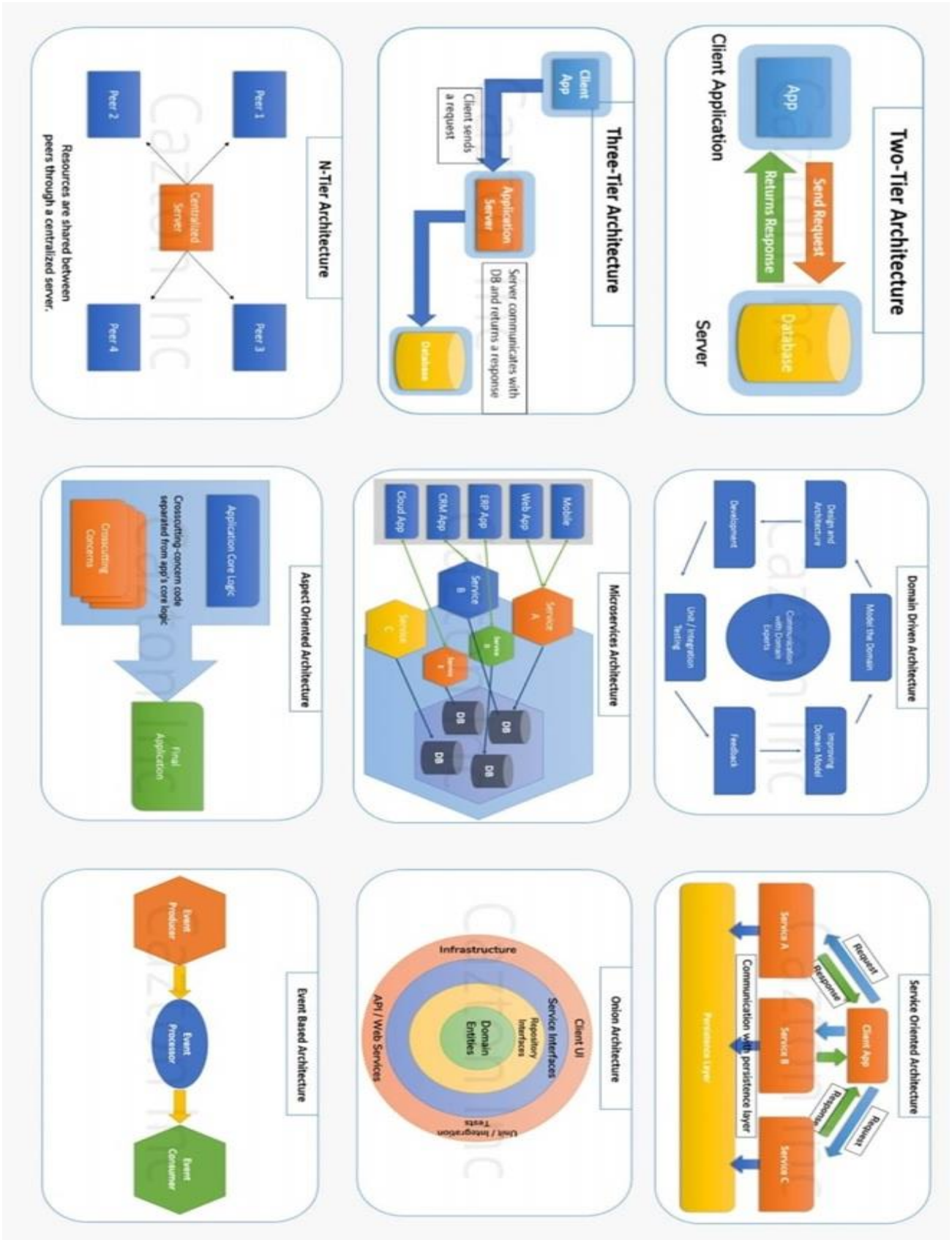
  Egyfajta szabványosított keretrendszer, olyan alapvető modell, amely leírja egy adott terület vagy rendszer alapvető jellemzőit, struktúráját és összetevőit. A referencia modell általában nem részletezi a terület vagy rendszer minden részletét, hanem a fő összetevőket és azok kapcsolódását írja le.

- Architektúra mintázat

  Olyan általános tervezési megoldások, amelyeket ismételten alkalmazhatunk a szoftvertervezés során.

- Alkalmazás kontextus

  Az alkalmazás külső környezete, amivel kommunikációt folytat, ami hatással van a működésére.



Szoftver Architektúra Minták

5.4.2 Szoftver Architektúra Minta (Architecture Pattern)

A szoftverarchitektúra-mintázatok olyan sablonok vagy tervezési elvek, amelyek segítenek az architektúra kialakításában és a szoftverkomponensek és rendszerek összekapcsolásában.

Segítségükkel az architektúra egyszerűbbé, karbantarthatóbbá és skálázhatóbbá válik.

Szoftver Architektúra mintázatok

MVC (Model-View-Controller)

Az MVC mintázat a felhasználói felületek tervezésének egyik legelterjedtebb megközelítése, ami szétválasztja az alkalmazás adatmodelljét (Model), a felhasználói felületet (View) és az alkalmazás irányítását (Controller).

Szervertől független állapot (Stateless)

Az alkalmazás állapota teljes mértékben a kliens oldalon van, ez lehetővé teszi a könnyű skálázhatóságot és a magas rendelkezésre állást.

Mikroszolgáltatások (Microservices)

Olyan alkalmazások fejlesztési módszere, amelyek a rendszert különálló, önállóan karbantartható kis szolgáltatásokra osztják fel.

Rétegzett architektúra (Layered Architecture)

Az alkalmazást különböző rétegekre osztják fel, például az adatrétegre, az üzleti logikai rétegre és a felhasználói felületre. Ez lehetővé teszi a komponensek könnyű cseréjét és a rendszer egyszerűbb karbantarthatóságát.

Üzenetközvetítés (Message Broker)

Az Üzenetközvetítés mintázat az adatok továbbítását üzeneteken keresztül valósítja meg, amelyeket egy központi üzenetközvetítő vagy busz kezel. Ez lehetővé teszi a független komponensek egyszerű és hatékony kommunikációját.

Repository

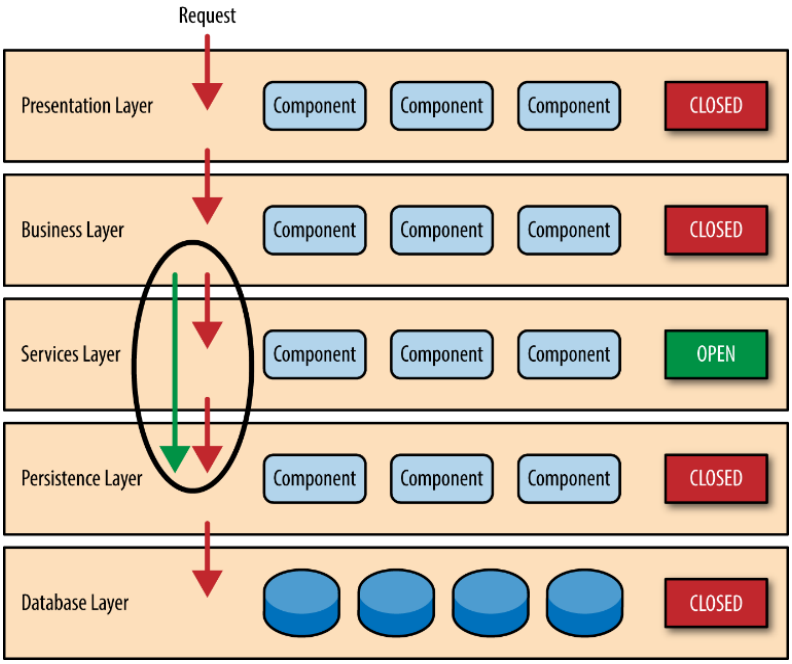
Segít az adatok kezelésében és hozzáférésében. Az adatokhoz hozzáférést biztosító központi osztályokat használja, amelyek egységes felületet biztosítanak az adatok kezeléséhez. Csökkenti a redundancia és az ismétlődés lehetőségét.

Publish-Subscribe

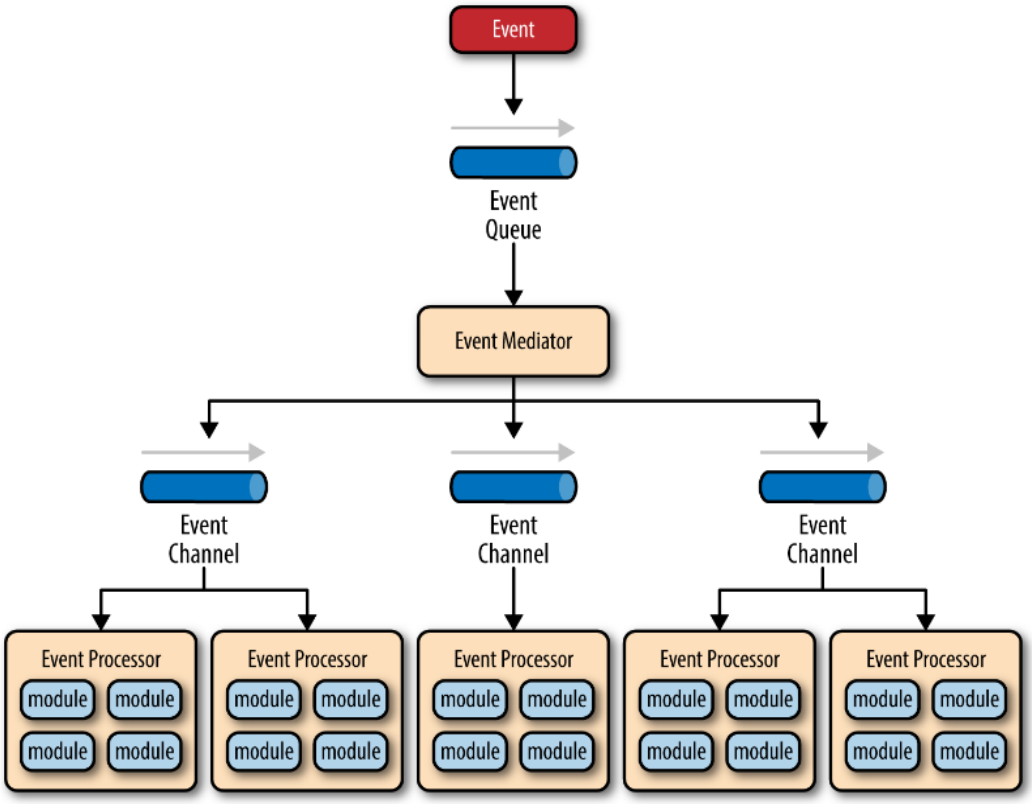
Segít az alkalmazások és rendszerek eseményalapú kommunikációjában. Lehetővé teszi az események publikálását (kiadását) és azok feliratkozókhoz történő továbbítását. Általában hatékonyabb és skálázhatóbb, mint a hagyományos, kéréseken alapú kommunikáció.

Microservices

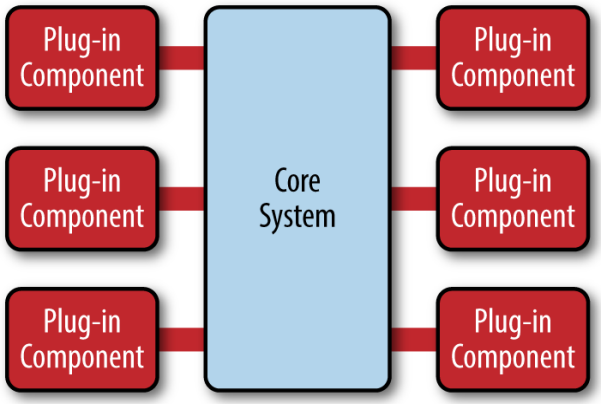
Az alkalmazást kisebb, önállóan működő egységekre bontja. Ezeket az egységeket mikroszolgáltatásoknak nevezi, és minden szolgáltatás saját folyamatban fut. A Microservices minta lehetővé teszi a könnyebb skálázhatóságot, a jobb karbantarthatóságot és a nagyobb agilitást a fejlesztők számára.



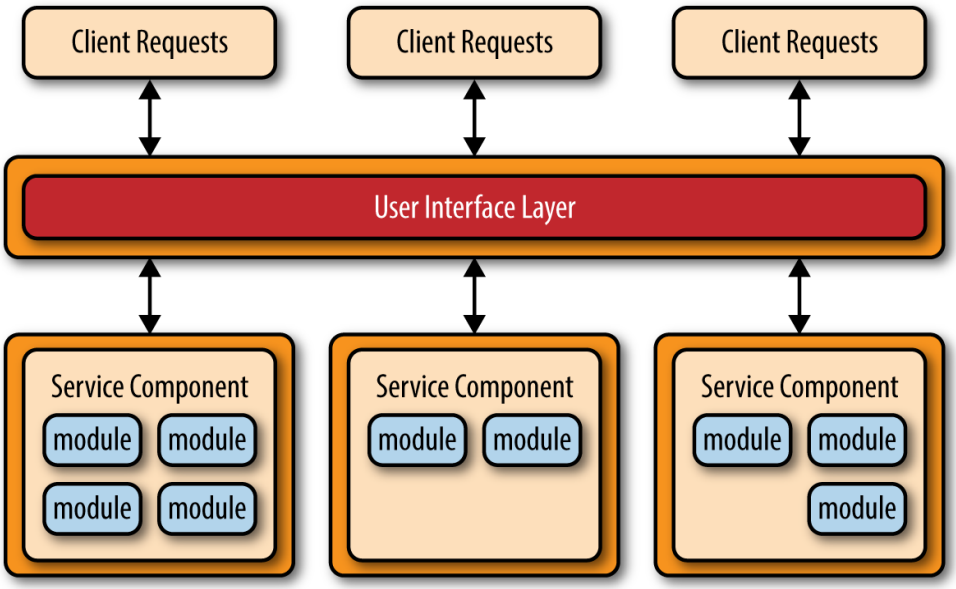
Rétegzett architektúra



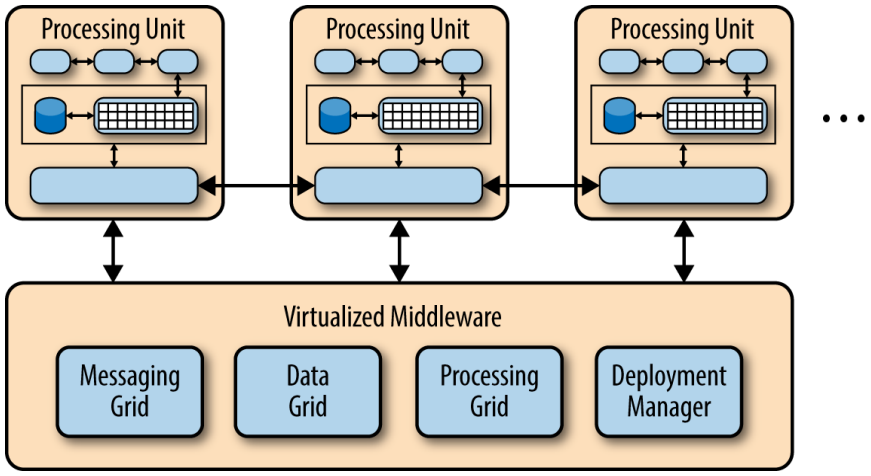
Esemény vezérelt architektúra



Mikrokernel architektúra



Mikroszolgáltatás architektúra



Téralapú architektúra

5.4.3 Szoftver Architektúra - Mikroszolgáltatás

A Mikroszolgáltatás architektúra egy szoftvertervezési megközelítés, független szolgáltatásokból álló alkalmazások laza rendszere, ahol az egyes alkalmazások önálló szolgáltatás modulokkal kezelik az üzleti logikát, az adatmanipulációt, a konfigurációt, a tárolást és az alkalmazás egyéb összetevőit. A Mikroszolgáltatások alkalmazása lehetővé teszi a nagy rendszerek fejlesztését és karbantartását kisebb, egyszerűbb és könnyebben kezelhető részekre bontva.

Az egyes Mikroszolgáltatásokat a lehető leginkább egymástól függetlenül kell megtervezni és fejleszteni, azonban az egész rendszer szempontjából összhangban kell működniük, és a kommunikációjukat hatékonyan kell megoldani. Az egyes szolgáltatásokat gyakran különböző programnyelvekben és technológiákban valósítják meg, így a fejlesztőknek különböző szaktudásokkal kell rendelkezniük.

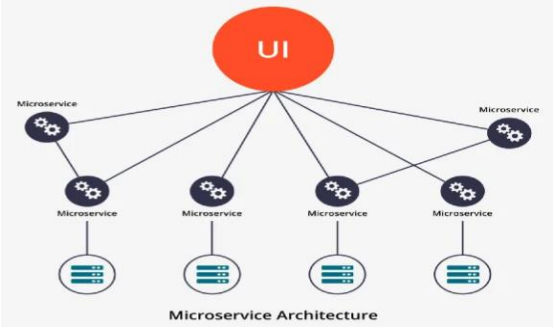
Előnyök

- Erős skálázhatóság
- Könnyebb szervezés és menedzselés
- Modulfüggetlenül fejlesztés
- Technológia semlegesség (agnosztikus)

További előnye, hogy az egyes szolgáltatások fejlesztése és karbantartása egymástól függetlenül történhet, így a hibajavítás és a frissítések is könnyebben kezelhetőek.

Hátrányok

- Erősen tervezés centrikus
- Nehezebb tesztelhetőség
- Biztonság érzékenység
- Összetettség
- Nagyobb infrastrukturális költségek
- Különféle szolgáltatások közötti kommunikáció nehezítettsége

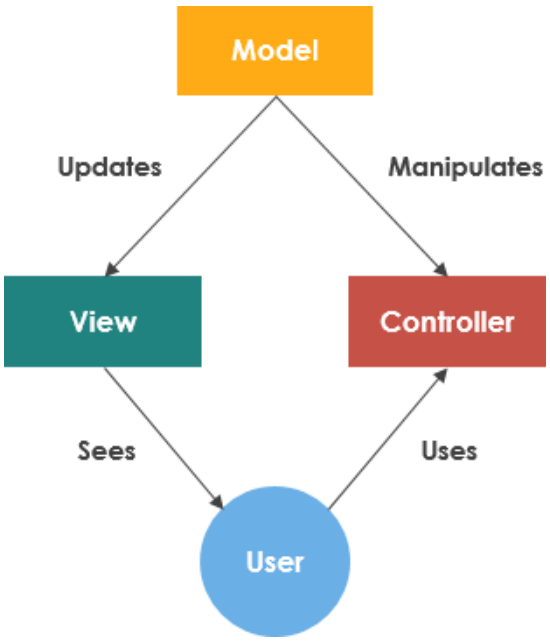


5.4.4 Szoftver Architektúra – Model-View-Controller (MVC)

Architekturális tervezési minta, amely a problémák szétválasztásával javítja az alkalmazásszervezést. Az interaktív alkalmazást három komponensre osztja: Modell / nézet és vezérlő. Az üzleti adatok (modellek) és a felhasználói felületek (nézetek) elkülönítését kényszeríti ki, egy harmadik komponens (vezérlők) pedig hagyományosan a logikát, a felhasználói bemenetet kezeli, és koordinálja a modelleket és a nézeteket.

Az MVC célja, hogy segítsen strukturálni az alkalmazás problematikájának három részre történő szétválasztását:

- A **modell** felelős az alkalmazás adatainak kezeléséért. A vezérlőtől kapja a felhasználói bemenetet.
- A **nézet** a modell egy adott formátumban történő megjelenítését jelenti.
- A **vezérlő** reagál a felhasználói bemenetre, és interakciókat hajt végre az adatmodell objektumain. A vezérlő fogadja a bemenetet, opcionálisan érvényesíti azt, majd továbbítja a bemenetet a modellnek.



Az MVC keretrendszer célja

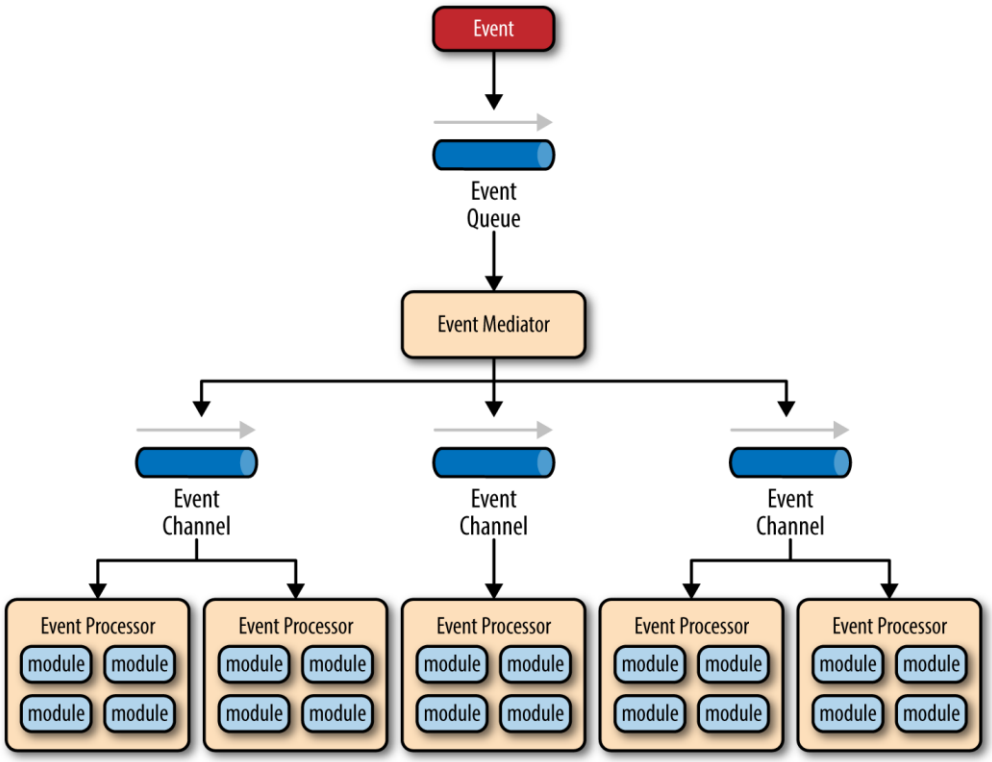
- Egyidejű fejlesztés - Mivel az MVC szétválasztja az alkalmazás különböző komponenseit, a fejlesztők párhuzamosan dolgozhatnak a különböző komponenseken anélkül, hogy befolyásolnák vagy blokkolnák egymást.
- Skálázhatóság - ha az alkalmazás teljesítményproblémákkal küzd, mert az adatbázishoz való hozzáférés lassú, akkor az adatbázist futtató hardvert frissíthető anélkül, hogy ez más komponenseket érintene.
- Laza csatolás (függés) - az MVC keretrendszer természete miatt a modellek, nézetek és vezérlők között alacsony a csatolás.
- Bővíthetőség - mivel a komponensek kis mértékben függenek egymástól, az egyik komponens módosítása (a hibák kijavítása vagy a funkcionalitás megváltoztatása érdekében) nem érinti a másikat.

5.4.5 Szoftver Architektúra – Esemény vezérelt (EDA)

A rendszer komponensei egymással események (event) által kommunikálnak. Az események általában olyan jelzések, amelyek valamilyen jelentős eseményt jelölnek, például egy tranzakció végét, vagy egy adatbázis rekordjának módosulását. Az EDA azt hangsúlyozza, hogy a rendszert az események állapotának változása hajtja előre, nem pedig a folyamatok.

Az események általában egy központi eseményközpont (event hub) által kezeltek, amely továbbítja őket a megfelelő rendszerelemekhez. Az eseményközpont használhat aszinkron kommunikációt, így az egyes rendszerelemek nem blokkolják egymást a kommunikáció során. Az események kezelésére olyan szolgáltatásokat használnak, mint az eseményfeldolgozó rendszerek, az esemény vezérelt adatfeldolgozás vagy az eseményalapú integráció.

Az esemény vezérelt architektúra előnyei közé tartozik a nagyobb skálázhatóság, az erőforrások hatékonyabb kihasználása és a rendszerelemek függetlenítése egymástól. Az esemény vezérelt architektúra lehetővé teszi a rendszer rugalmasabb felépítését, könnyebb karbantarthatóságát és a jobb hibatűrést, mivel az egyes rendszerelemek függetlenül működnek egymástól. Az esemény vezérelt architektúra a modern, nagy adatvolumenű rendszerek kialakításában egyre gyakrabban alkalmazott megközelítés.



5.5 Infrastruktúra Architektúra

5.5.1 Hardware tervezés

Az alkalmazások hardware szükségleteinek tervezése során az alábbi lépéseket kell követni:

Az alkalmazás követelményeinek megértése: Az alkalmazás szükségleteinek megértése az első lépés. Ezt általában az alkalmazás funkcionális és nem funkcionális követelményeinek felmérése és az alkalmazás által elvárt teljesítmény és rendelkezésre állás megértése jelenti.

Rendszerarchitektúra tervezése: Az alkalmazás hardware szükségleteinek tervezése során az alkalmazás rendszerarchitektúráját is meg kell tervezni. Ennek során meg kell határozni a rendszer elemeit, például a processzorokat, a memóriát, a tárolóeszközöket, a hálózati kapcsolatokat és az egyéb hardverkomponenseket.

Hardverelemek kiválasztása: Az egyes hardverelemek kiválasztása az alkalmazás szükségleteitől függ. A processzorok, memóriák, tárolóeszközök és hálózati kapcsolatok kiválasztása során figyelembe kell venni az alkalmazás követelményeit és az alkalmazás által elvárt teljesítményt.

Skálázhatóság megtervezése: Az alkalmazás hardware szükségleteinek tervezése során fontos a skálázhatóság figyelembevétele is. Az alkalmazásnak lehetővé kell tennie a számítási erő és a tárolókapacitás dinamikus növelését, ha az alkalmazás használata bővül.

Biztonsági intézkedések: Az alkalmazás biztonságának garantálása is fontos tényező az alkalmazás hardware szükségleteinek tervezésekor. Az alkalmazásnak biztosítania kell a biztonsági mentési lehetőséget és a redundanciát az adatvesztés elkerülése érdekében.

Az alkalmazás hardware szükségleteinek tervezése során a fenti lépéseket kell követni, hogy biztosítsuk az alkalmazás hatékony és megbízható működését. A megfelelő hardware szükségletek kiválasztása és a megfelelő rendszerarchitektúra tervezése lehetővé teszi a hatékonyabb és megbízhatóbb alkalmazásfejlesztést.

5.5.2 Adatbázis környezet jellemzők

A nagyméretű adatbázisok jellemzően nagy mennyiségű adatot tárolnak és kezelnek, így az ilyen adatbázisokhoz szükséges hardware igények a következők:

Magas tárolókapacitás: A nagyméretű adatbázisokhoz magas tárolókapacitás szükséges. A tárolókapacitás lehetővé teszi az adatbázisok nagy mennyiségű adat tárolását és hatékony kezelését.

Nagy memória: A nagy memória lehetővé teszi az adatbázisok hatékonyabb kezelését és gyorsabb lekérdezéseit.

Több processzor: A több processzor lehetővé teszi az adatbázisok nagyobb adatmennyiségek és felhasználói igények hatékonyabb kezelését.

Magas sávszélességű hálózati kapcsolatok: A magas sávszélességű hálózati kapcsolatok lehetővé teszik az adatbázisok gyors átvitelét a különböző szerverek között.

Automatikus mentés és redundancia: A nagyméretű adatbázisokhoz automatikus mentési lehetőségek és redundancia szükségesek, amelyek biztosítják az adatok folyamatos védelmét és hozzáférhetőségét.

Magas rendelkezésre állás: A nagyméretű adatbázisok esetében magas rendelkezésre állás szükséges, hogy az adatbázis mindig elérhető legyen és a felhasználók hatékonyan használhassák.

Gyors adatelérés: A nagyméretű adatbázisok esetében a gyors adatelérés fontos igény, hogy a felhasználók hatékonyan tudják használni az adatbázist.

Az állandó fejlesztéseknek köszönhetően a hardware technológiák és a rendszerkomponensek egyre hatékonyabbak lesznek, amely lehetővé teszi a nagyobb adatmennyiségek hatékonyabb kezelését és feldolgozását.

5.5.2.1 Párhuzamos feldolgozás környezet jellemzők

A párhuzamos feldolgozás célja a számítógépes feladatok hatékonyabb elvégzése több processzor vagy számítógép egyidejű használatával. A párhuzamos feldolgozás megvalósításához szükséges hardware tervezési jellemzők a következők:

Többmagú processzorok: A többmagú processzorok lehetővé teszik a számítógépes feladatok hatékonyabb elvégzését, mivel több feladatot képesek egyszerre futtatni.

Számítógép hálózatok: A számítógép hálózatok lehetővé teszik a számítógépek egyidejű használatát, amely lehetővé teszi a nagyobb adatmennyiségek és feladatok hatékonyabb feldolgozását.

Memória: A nagyobb memória lehetővé teszi a nagyobb adatmennyiségek és feladatok hatékonyabb kezelését.

Tárolókapacitás: A nagyobb tárolókapacitás lehetővé teszi a nagyobb adatmennyiségek hatékonyabb kezelését és tárolását.

Magas sebességű hálózati kapcsolatok: A magas sebességű hálózati kapcsolatok lehetővé teszik a nagyobb adatmennyiségek gyors átvitelét a számítógépek között.

Párhuzamosított algoritmusok: A párhuzamos feldolgozás hatékonyabbá tételéhez a feladatokat olyan algoritmusokkal kell megtervezni, amelyek képesek a párhuzamos feldolgozásra.

Párhuzamos feldolgozásra képes operációs rendszer: A párhuzamos feldolgozás során az operációs rendszernek képesnek kell lennie a feladatok párhuzamos végrehajtására és az erőforrások hatékony kezelésére.

Az állandó fejlesztéseknek köszönhetően a hardware technológiák és a rendszerkomponensek egyre hatékonyabbak lesznek, amely lehetővé teszi a nagyobb adatmennyiségek és feladatok hatékonyabb kezelését és feldolgozását.

5.5.2.2 Elosztott alkalmazás környezet jellemzők

Az elosztott alkalmazások hardware igényei az elosztott rendszer megvalósításától és az alkalmazások igényeitől függenek. Az alábbiakban néhány általános hardware igényt sorolok fel, amelyek jellemzőek az elosztott alkalmazásokra:

Több szerver: Az elosztott alkalmazások általában több szerveren futnak, így a rendszer erőforrásait megosztva lehetővé teszik az alkalmazások folyamatos elérhetőségét és az adatok hatékony kezelését.

Skálázhatóság: Az elosztott alkalmazások lehetővé teszik az erőforrások dinamikus skálázását, amely lehetővé teszi a rendszer kapacitásának növelését a megnövekedett terhelés kezeléséhez.

Nagy tárolókapacitás: Az elosztott alkalmazások gyakran nagy mennyiségű adatot dolgoznak fel, így a nagy tárolókapacitás fontos igény. A nagy tárolókapacitás lehetővé teszi a nagy mennyiségű adatok hatékony kezelését és tárolását.

Magas sávszélességű hálózati kapcsolatok: Az elosztott alkalmazások hatékony működéséhez magas sávszélességű hálózati kapcsolatok szükségesek. A magas sávszélességű hálózati kapcsolatok lehetővé teszik az adatok gyors átvitelét a különböző szerverek között.

Biztonság: Az elosztott alkalmazások esetében a biztonság kiemelt fontossággal bír. A rendszernek biztosítania kell a megfelelő biztonsági mentési lehetőséget és redundanciát az adatvesztés elkerülése érdekében, valamint a felhasználói adatok védelmét is.

Az elosztott alkalmazások esetében az alkalmazások és a rendszerkomponensek általában függetlenül működnek egymástól, és több szerveren futnak. Az állandó fejlesztéseknek köszönhetően a hardware technológiák és a rendszerkomponensek egyre hatékonyabbak lesznek, amely lehetővé teszi az elosztott alkalmazások hatékonyabb működését és nagyobb adatmennyiségek kezelését.

5.5.2.3 Felhő (Cloud) alapú környezet jellemzők

A felhő alapú megoldások hardware igényei az adott felhőszolgáltatástól függenek, de általánosságban a következők tartoznak hozzájuk:

Több szerver: A felhő alapú megoldások több szerveren futnak, így a rendszer erőforrásait megosztva lehetővé teszik az alkalmazások folyamatos elérhetőségét és az adatok hatékony kezelését.

Skálázhatóság: A felhő alapú megoldások lehetővé teszik az erőforrások dinamikus skálázását, amely lehetővé teszi a rendszer kapacitásának növelését a megnövekedett terhelés kezeléséhez.

Nagy tárolókapacitás: A felhő alapú megoldások nagy tárolókapacitást igényelnek a nagy mennyiségű adatok tárolásához.

Magas sávszélességű hálózati kapcsolatok: A felhő alapú megoldások hatékony működéséhez magas sávszélességű hálózati kapcsolatok szükségesek. A magas sávszélességű hálózati kapcsolatok lehetővé teszik az adatok gyors átvitelét a különböző szerverek között.

Biztonság: A felhő alapú megoldások esetében a biztonság kiemelt fontossággal bír. A rendszernek biztosítania kell a megfelelő biztonsági mentési lehetőséget és redundanciát az adatvesztés elkerülése érdekében, valamint a felhasználói adatok védelmét is.

Magas rendelkezésre állás: A felhő alapú megoldások esetében magas rendelkezésre állás szükséges, hogy az alkalmazások és az adatok mindig elérhetők legyenek és a felhasználók hatékonyan használhassák őket.

Automatikus skálázás: A felhő alapú megoldások automatikus skálázási lehetőségekkel rendelkeznek, amelyek lehetővé teszik a rendszer kapacitásának dinamikus növelését a megnövekedett terhelés kezeléséhez.

Az állandó fejlesztéseknek köszönhetően a hardware technológiák és a rendszerkomponensek egyre hatékonyabbak lesznek, amely lehetővé teszi a felhő alapú megoldások hatékonyabb működését és nagyobb adatmennyiségek kezelését.

5.5.3 Technológiák

A hardware technológiák olyan hardverelemek és hardvermegoldások, amelyeket a számítógépek és más elektronikai eszközök működéséhez használnak. A hardware technológiák folyamatosan fejlődnek, hogy nagyobb sebességgel, hatékonysággal és megbízhatósággal működjenek, és a számítástechnika és más elektronikai területeken használják őket.

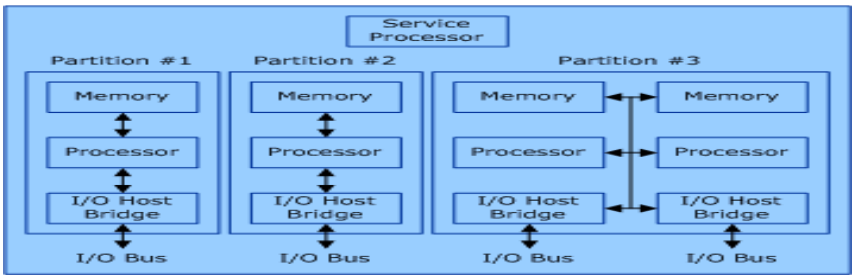
Hardware technológiák

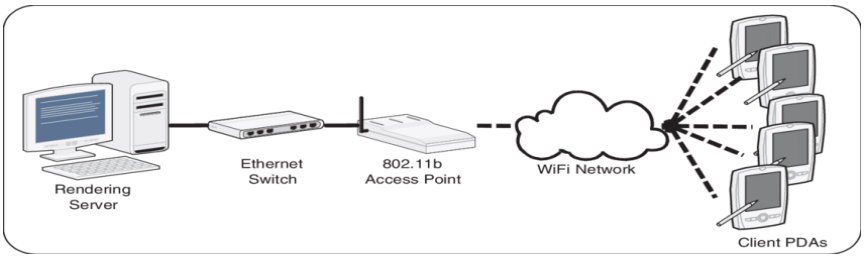
Processzorok: A processzorok az eszközök "agyát" jelentik, és a számítógépek és más elektronikai eszközök működéséhez szükségesek. Az újabb processzorok több maggal rendelkeznek, és magasabb sebességgel és hatékonysággal működnek.

Memória: A memória az eszközökben tárolt adatokat és programokat jelenti. Az újabb memóriák nagyobb kapacitással és sebességgel rendelkeznek, hogy lehetővé tegyék a nagyobb adatmennyiség kezelését.

Tárolóeszközök: A tárolóeszközök, például merevlemezek, SSD-k és flash-meghajtók, az adatok hosszú távú tárolására szolgálnak. Az újabb tárolóeszközök nagyobb tárolókapacitással és sebességgel rendelkeznek, hogy lehetővé tegyék a nagyobb adatmennyiség tárolását és hozzáférését.

Videokártyák: A videokártyák a számítógépekben azon dolgoznak, hogy a képernyőn megjelenített grafikai információk a lehető legjobb minőségben és sebességgel jelenjenek meg. Az újabb videokártyák nagyobb grafikus kapacitással rendelkeznek, és lehetővé teszik a nagyobb felbontású és magasabb képfrissítésű játékok és videók megjelenítését.





5.5.4 Teljesítmény méretezés

A hardware teljesítmény méretezés azon folyamat, amely során a rendszer különböző elemeit, például a processzort, a memóriát és a tárolóeszközöket úgy választjuk meg, hogy a rendszer megfelelően teljesítse a meghatározott feladatokat.

A hardware teljesítmény méretezés jellemzői

CPU: A processzor teljesítményének méretezése az egyik legfontosabb tényező a hardware teljesítmény méretezés során. A processzorok sebessége, magok száma és gyártási technológiája határozza meg a teljesítményüket.

Memória: A rendszer memóriájának kapacitása és sebessége fontos tényező az alkalmazások és folyamatok hatékony működése szempontjából. A nagyobb memória kapacitás lehetővé teszi a nagyobb adatmennyiségek kezelését, míg a magasabb memóriasebesség növeli a rendszer reakcióidejét.

Tárolóeszközök: A tárolóeszközök kapacitása és sebessége szintén fontos tényező a rendszer teljesítményének méretezésekor. A nagyobb kapacitás lehetővé teszi a nagyobb adatmennyiség tárolását, míg a magasabb sebesség lehetővé teszi az adatok gyorsabb olvasását és írását.

Hálózati kapcsolatok: A rendszer hálózati kapcsolatainak sebessége és kapacitása meghatározó a nagy adatforgalmat igénylő alkalmazások és folyamatok esetében. A magasabb sebességű és kapacitású hálózati kapcsolatok lehetővé teszik a nagyobb adatmennyiségek hatékony átvitelét a rendszerben.

Alkalmazások igényei: A hardware teljesítmény méretezésekor az alkalmazások és folyamatok igényei is figyelembe kell venni. Az egyes alkalmazások és folyamatok eltérő igényekkel rendelkeznek a processzor teljesítményére, memória kapacitására és tárolókapacitására vonatkozóan.

Az állandó fejlesztéseknek köszönhetően a hardware technológiák és a rendszerkomponensek egyre hatékonyabbak lesznek, amely lehetővé teszi a nagyobb adatmennyiségek kezelését és az alkalmazások hatékonyabb futtat

5.5.5 Skálázás

A hardware skálázhatóság tervezés jellemzői

Horizontális és vertikális skálázhatóság: A rendszer skálázhatósága lehet horizontális, azaz a rendszer bővítése azonos hardverelemek hozzáadásával, vagy vertikális, azaz a rendszer bővítése erősebb hardverelemek hozzáadásával.

Felhasználói igények: A skálázhatósági tervezés során figyelembe kell venni az alkalmazás igényeit és a felhasználói igényeket is, amelyeket a rendszernek kiszolgálnia kell. Például, ha az alkalmazás megnövekedett felhasználói terhelést jelent, akkor a rendszernek képesnek kell lennie a folyamatok dinamikus bővítésére.

Adatbázis mérete: Az adatbázis mérete is meghatározó tényező a rendszer skálázhatósági tervezésekor. A nagyobb adatbázisok kezelése nagyobb kapacitást igényel a rendszertől.

Kliens oldal: A rendszer skálázhatósági tervezésénél fontos figyelembe venni a kliens oldalt is, és biztosítani a szükséges kapacitást és teljesítményt a kliensoldali készülékek számára.

Biztonság: A rendszer skálázhatósági tervezésekor a biztonsági kérdések is fontosak. A rendszernek biztosítania kell a megfelelő biztonsági mentési lehetőséget és redundanciát az adatvesztés elkerülése érdekében.

Hardverelemek választása: A rendszer skálázhatósági tervezésekor az egyes hardverelemek, például a processzorok, memória, tárolóeszközök és hálózati kapcsolatok kiválasztása is fontos tényező. A nagyobb kapacitású és erősebb hardverelemek lehetővé teszik a rendszer nagyobb terhelésének kezelését és a nagyobb adatmennyiségek hatékonyabb kezelését.

Az állandó fejlesztéseknek köszönhetően a hardware technológiák és a rendszerkomponensek egyre hatékonyabbak lesznek, amely lehetővé teszi a nagyobb adatmennyiségek kezelését és az alkalmazások hatékonyabb futtatását. A megfelelő hardware skálázhatósági tervezés lehetővé teszi a rendszer hatékonyabb és megbízhatóbb működés

5.5.6 Redundancia

A hardware redundancia tervezés jellemzői

Redundáns rendszerek: A redundancia tervezése során általában redundáns rendszerek kiépítése történik, amelyek biztosítják az adatok és folyamatok folyamatos elérhetőségét. Az ilyen rendszerek esetén a rendszer bizonyos elemei kettős végrehajtást kapnak, így, ha egy rendszerkomponens meghibásodik, a másik automatikusan átveszi a feladatot.

Többpontos kapcsolatok: A redundancia tervezése során fontos figyelembe venni a hálózati kapcsolatokat is. A többpontos kapcsolatok kiépítése és a hálózati rendszer megfelelő konfigurálása lehetővé teszi a hálózat folyamatos elérhetőségét, még akkor is, ha az egyes elemek meghibásodnak.

Adatmentési lehetőségek: A redundancia tervezésekor fontos az adatmentési lehetőségek biztosítása is. A biztonsági mentési lehetőségek és a redundancia lehetővé teszi az adatvesztés elkerülését és a rendszer megbízhatóbb működését.

Hardverelemek választása: A redundancia tervezésekor az egyes hardverelemek, például a processzorok, memória, tárolóeszközök és hálózati kapcsolatok kiválasztása is fontos tényező. A megfelelő hardverelemek biztosítják a rendszer megfelelő redundanciáját és az adatok folyamatos elérhetőségét.

Automatikus helyreállítás: A redundancia tervezésekor fontos figyelembe venni az automatikus helyreállítást is. Az automatikus helyreállítási folyamatok lehetővé teszik a rendszer hibaelhárítását és az adatok folyamatos elérhetőségét, még akkor is, ha egy rendszerkomponens meghibásodik.

A hardware redundancia tervezése lehetővé teszi a rendszer megbízhatóbb működését és az adatok folyamatos elérhetőségét, még akkor is, ha egy rendszerkomponens meghibásodik. Az állandó fejlesztéseknek köszönhetően a hardware technológiák és a rendszerkomponensek egyre megbízhatóbbak és hatékonyabbak lesznek, amely lehetővé teszi a nagyobb adatmennyiségek kezelését és az alkalmazások hatékonyabb futtatását.

5.6 Rendszertervezés

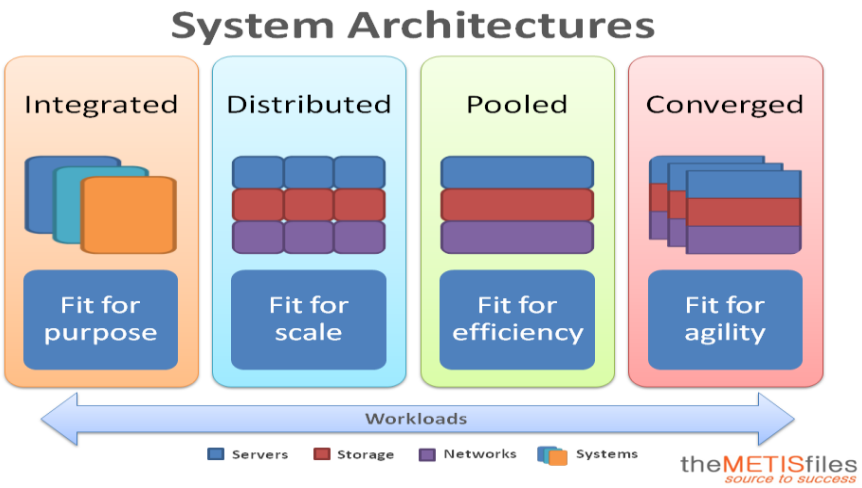
Az IT rendszertervezés olyan folyamat, amely során a vállalat üzleti igényeit és az azokat támogató informatikai megoldásokat egyesítik. Az IT rendszertervezés során a tervező megvizsgálja a vállalat üzleti folyamatait, az információáramlást és az erőforrásokat annak érdekében, hogy olyan informatikai rendszert hozzon létre, amely hatékonyan támogatja a vállalat céljait.

Ez a folyamat magába foglalja a rendszer céljainak, követelményeinek és korlátainak meghatározását, az architektúra tervezését, a rendszer komponenseinek, funkcióinak és folyamatainak meghatározását, valamint a rendszer kialakítását és implementálását.

A rendszertervezés fontos elemei közé tartozik az adatok és az információk kezelése, az adatbázis tervezése, a rendszer integrációja más rendszerekkel, a rendszer biztonságának megtervezése és a felhasználói felület tervezése.

A rendszertervezés során többféle tervezési modellt használható, például a vízesés modellt, az agilis modellt, a prototípus modellt vagy a spiral modellt. A választott modell az adott projekt jellegétől, a tervezők és fejlesztők tapasztalatától és a rendszerre vonatkozó követelményektől függ.

A rendszertervezésnek fontos szerepe van a rendszer hatékony működésének biztosításában, és elengedhetetlen azokban az esetekben, amikor komplex rendszerek tervezése és implementálása szükséges, például nagyvállalati rendszerek, adatbázisok, informatikai infrastruktúrák vagy telekommunikációs rendszerek tervezésekor.



5.6.1 Tervezési módszertan

Tervezési elképzelések kiindulópontjai

- **Követelmények**
- **Technológiai környezet**
- **(tervezői) Tapasztalat**

Igényfelmérés: A tervezők az üzleti igényeket és elvárásokat veszik figyelembe. Az üzleti folyamatok elemzése segít a tervezőknek abban, hogy meghatározzák, hogy milyen funkciókra van szükségük az informatikai rendszerben.

Tervezés: Az üzleti igények és funkciók alapján a tervezők elkészítik az informatikai rendszer tervét. Ez magában foglalja a szükséges szoftverek, hardverek és adatbázisok kiválasztását.

Megvalósítás: A tervezés után a rendszer megvalósításra kerül. Ez magában foglalja a szükséges szoftverek és hardverek telepítését, a szükséges adatok átvitelét és a rendszer konfigurálását.

Tesztelés: Miután az informatikai rendszer elkészült, a tervezők tesztelik, hogy az minden szükséges funkciót támogat és a megfelelően működik-e.

Üzemeltetés és karbantartás: Az informatikai rendszer karbantartása, frissítése és javítása az életciklusának következő szakasza.

Az IT rendszertervezés során fontos, hogy **a tervező ne csak a jelenlegi igényekre koncentráljon, hanem előre is gondolkodjon** és olyan rendszert tervezzen, amely rugalmasan alkalmazkodik a jövőbeni igényekhez is.

5.6.2 Tervezési hibatípusok

Az alábbi hiba típusok hosszútávú használat során jelentkező problémákat okozhatnak :

- **Elavult rendszerek fenntartása**
- **Hiányzó dokumentáció**
- **Hiányzó tesztelés**

5.6.3 Szoftver tervezés típusok

A tervezési típusok olyan általános tervezési megközelítések, amelyeket lehetővé teszik a tervezési folyamat hatékonyabbá.

Tervezési típusok

Strukturális tervezés: A strukturális tervezés célja a rendszer vagy alkalmazás adatmodelljének létrehozása és a rendszer vagy alkalmazás komponenseinek strukturális összehangolása.

Funkcionális tervezés: A funkcionális tervezés célja a rendszer vagy alkalmazás kívánt funkcionalitásának meghatározása és a rendszer vagy alkalmazás feladatokra bontása.

Objektumorientált tervezés: Az objektumorientált tervezés célja az alkalmazások és rendszerek objektumokra alapozott struktúrájának létrehozása és a rendszer vagy alkalmazás objektumorientált tervezési elveinek betartása.

Adatelemző tervezés: Az adatelemző tervezés célja a rendszer vagy alkalmazás adatmodelljének létrehozása, az adatok tárolási és karbantartási eljárásainak meghatározása.

Rendszerszintű tervezés: A rendszerszintű tervezés célja a rendszer vagy alkalmazás komponenseinek és azok közötti kapcsolatoknak a tervezése és optimalizálása.

A tervezési típusok különböző szempontokat vesznek figyelembe a tervezés során, és lehetővé teszik a tervezési folyamat hatékonyabbá tételét az alkalmazások és rendszerek hatékonyabb megvalósításához.

5.7 Szoftverfejlesztés

A szoftverfejlesztés az a folyamat, amely során szoftvereket terveznek, fejlesztenek, tesztelnek, telepítenek és karbantartanak. A szoftverfejlesztés célja az üzleti igények kielégítése és a felhasználók számára hasznos és értékes szoftverek létrehozása.

5.7.1 Alapelvek

- **YAGNI**: "You Ain't Gonna Need It" (Nem lesz rá szükséged !). Ez egy tervezési elv a szoftvertervezésben, amely azt javasolja, hogy ne adjunk hozzá funkciókat a programhoz, amelyeket nem kérnek, vagy amelyekre jelenleg nincs szükség.
- **SOLID**: Az öt SOLID elv, amelyek az Objektumorientált tervezésben (OOP) segítenek a rugalmas és karbantartható kód készítésében:
    - **Single Responsibility Principle** (SRP): Egy osztálynak csak egy felelőssége legyen.
    - **Open/Closed Principle** (OCP): Az osztályoknak nyitottnak kell lenniük a kiterjesztésre, de zártaknak a módosításokra.
    - **Liskov Substitution Principle** (LSP): Az al-osztályoknak helyettesíthetőnek kell lenniük a feljebbvaló osztályokban.
    - **Interface Segregation Principle** (ISP): Az interfészeknek csak olyan metódusokat kell tartalmazniuk, amelyeket az osztályok ténylegesen használnak.
    - **Dependency Inversion Principle** (DIP): Az osztályoknak függetlennek kell lenniük a más osztályok implementációjától, és csak az absztrakt interfészekre kell támaszkodniuk.
- **KISS**: "Keep It Simple, Stupid" (Ne bonyolítsd !). Ez az elv azt javasolja, hogy tartsuk egyszerűen a kódot és az architektúrát, hogy könnyen érthető és karbantartható legyen.
- **DRY**: "Don't Repeat Yourself" (Ne ismételd önmagad !). Ez az elv azt javasolja, hogy kerüljük a redundáns kódot és az ismétlődő műveleteket a programban, és helyette használjunk újra felhasználható modulokat és funkciókat.

5.7.2 Szoftverfejlesztési módszertan

A szoftverfejlesztési módszertanok olyan eljárásrendek, amelyeket a szoftverfejlesztés során alkalmaznak.

A szoftverfejlesztési módszertanok segítenek a fejlesztőknek abban, hogy strukturáltabb és hatékonyabb módon dolgozzanak, azáltal, hogy meghatározzák az eljárásrendet, amelyet követni kell a szoftverfejlesztési folyamat során.

Az egyes módszertanok saját szabályrendszerrel rendelkeznek, amelyek definiálják a folyamatokat, az eljárásokat és az egyes szereplők feladatait.

A szoftverfejlesztési módszertanok többféle típusba sorolhatók, és általában a fejlesztési folyamatok feladatainak, szereplőinek, életciklusának vagy jellegének alapján különböztetik meg őket.

Az életciklus fázis alapú típusok

- **lineáris** (a lépések szigorúan egymás után következnek, nincs visszalépés az előző fázisra),
- **spirális** (a lépések finomodva ismétlődnek),
- **iteratív vagy inkrementális** (prototípus készül, és később ezt finomítják ismétlődő lépések során át).

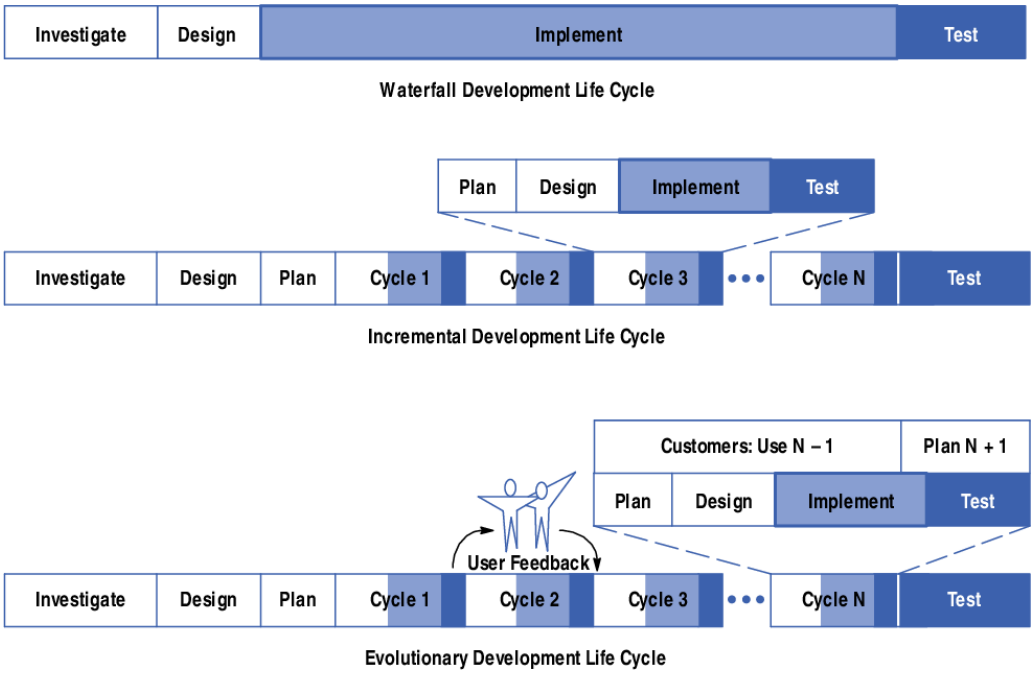
Dokumentáltság alapú típusok

- könnyűsúlyú (lightweight) módszertan, mely kevés dokumentum készítését írja elő,
- nehézsúlyú (heavyweight) módszertan, mely kimerítő dokumentumok készítését írja elő.

Koncepció alapú típusok

- adatközpontú
- folyamatközpontú
- követelményközpontú
- használatieset-központú
- tesztközpontú
- felhasználóközpontú
- emberközpontú
- csapatközpontú

Fig. 1. *Different models of the software development life cycle.*

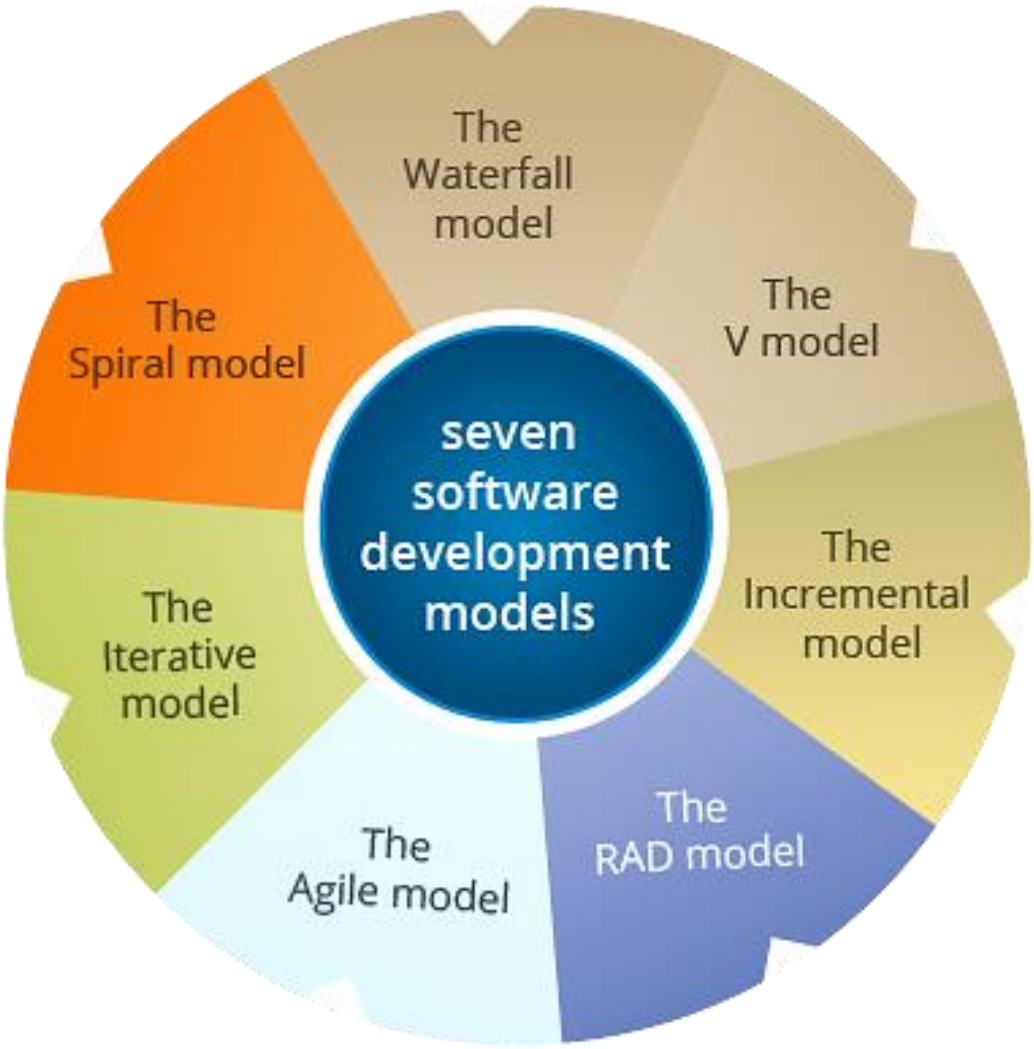


Módszertan példák

Vízesés módszer: A vízesés modell a szoftverfejlesztés hagyományos módszertana. Az eljárás egy lineáris, szekvenciális modellt alkalmaz, és egymást követő szakaszokat határoz meg, mint például az igényfelmérés, tervezés, fejlesztés, tesztelés és karbantartás. A módszer előnye a könnyen érthető folyamat és a szigorú dokumentáció. Az egyes szakaszok közötti átmenetek szigorúak és előre meghatározottak.

Agilis módszer: Az agilis módszer olyan szoftverfejlesztési módszer, amely a vízesés módszerrel szemben rugalmasabb. Egy iteratív és inkrementális folyamat, amely lehetővé teszi a fejlesztőknek, hogy a fejlesztés során rugalmasan reagáljanak a változásokra és az új követelményekre. Az agilis módszertanok között találhatók a Scrum, Kanban, XP (eXtreme Programming) és más módszertanok. Az agilis módszerek fő előnyei az állandó visszacsatolás, a gyors reakcióképesség és az ügyfélközpontúság.

DevOps módszer: A DevOps (Development and Operations) módszer az alkalmazásfejlesztési folyamat és az üzemeltetési folyamat integrált megközelítése. Az alkalmazások fejlesztése és üzemeltetése közötti határok elmosódása lehetővé teszi az alkalmazások és rendszerek gyorsabb kiadását, javítását és karbantartását.


The Waterfall model
The V model
The Incremental model
The RAD model
The Agile model
The Iterative model
The Spiral model
seven software development models

5.7.2.1 Iteratív szoftverfejlesztés

Az iteratív szoftverfejlesztés egy rugalmas szoftverfejlesztési módszertan, amely a fejlesztési folyamatot iteratív és ismétlődő ciklusokra bontja. Az iteratív szoftverfejlesztés célja a fejlesztési folyamat felgyorsítása, az ügyfél igényeinek jobb megértése és az alkalmazás minőségének javítása.

Az iteratív szoftverfejlesztés jellemzői

Rugalmasság: Az iteratív szoftverfejlesztés nagyobb rugalmasságot biztosít, mint a hagyományos szoftverfejlesztési módszertanok, mivel lehetővé teszi a folyamatok és a követelmények változtatását, és lehetővé teszi a fejlesztőknek, hogy a fejlesztési folyamatot alkalmazkodjanak az ügyfél igényeihez.

Ciklusok: Az iteratív szoftverfejlesztés a fejlesztési folyamatot rövid, ismétlődő ciklusokra bontja. Minden ciklusban az alkalmazás egy új verziója készül, amelyet az ügyfél vagy a tesztelők tesztelnek, és a visszajelzéseket az újabb ciklusban a követelményekhez illesztik.

Ügyfélközpontúság: Az iteratív szoftverfejlesztés nagyobb figyelmet szentel az ügyfél igényeinek megértésére, és az ügyfél visszajelzéseire épül. Az ügyfél bevonása a folyamatba lehetővé teszi, hogy az alkalmazás a valódi felhasználói igényeknek feleljen meg.

Gyakori tesztelés: Az iteratív szoftverfejlesztési módszertanban a tesztelés fontos szerepet játszik. Minden ciklusban a fejlesztők tesztelik az új verziót, hogy biztosítsák a funkcionális követelmények teljesítését, és lehetővé teszik a hibák korai felfedezését.

Inkrementális fejlesztés: Az iteratív szoftverfejlesztési módszertanban az alkalmazás fokozatosan épül fel, azaz az újabb ciklusokban újabb funkciók és jellemzők adódnak hozzá az előző verzióhoz. Az inkrementális fejlesztési módszertan segíti a fejlesztőket abban, hogy az alkalmazás fokozatosan bővüljön, és biztosítsa a funkciók hatékony működését

5.7.3 Szoftverfejlesztési paradigma

A szoftverfejlesztési paradigmák a szoftvertervezésben és fejlesztésben alkalmazott alapvető megközelítések és elméletek, amelyek meghatározzák a szoftverfejlesztési folyamatot, a fejlesztői csapatok szerepét és a szoftver architektúrát. A szoftverfejlesztési paradigmák határozzák meg, hogy hogyan kell felépíteni az alkalmazásokat és rendszereket, hogyan kell azokat tesztelni, és hogyan kell azokat karbantartani.

A **szoftverfejlesztési paradigmák** alapvetően három csoportra oszthatók:

- **strukturális**
- **objektumorientált**
- **funkcionális**

paradigma.

Strukturális paradigmák: A strukturális paradigmák a szoftverfejlesztés hagyományos megközelítésére épülnek, amely szerint a programot a feladatokat megoldó eljárások (funkciók) összetételével építjük fel. A strukturális paradigmák közül a legismertebb az eljárásközpontú paradigmája (procedural paradigm), amely a szoftverfejlesztést az eljárások (funkciók) szervezésére és az adatok közötti kapcsolatokra fókuszálja.

Objektumorientált paradigmák: Az objektumorientált paradigmák a szoftverfejlesztés újabb megközelítését jelentik, amely a programozást objektumokra alapozza. Az objektumorientált paradigmák közül a legismertebb a Java és a C++ nyelvek.

Funkcionális paradigmák: A funkcionális paradigmák a matematikai függvények használatára épülnek, és a szoftverfejlesztést olyan függvények sorozataként kezelik, amelyek paraméterekből állnak, és egy kimeneti értéket állítanak elő. A funkcionális paradigmák közül a legismertebb a Haskell és a Lisp.

Szoftverfejlesztési paradigmák

Imperatív paradigmák: Az imperatív paradigmák a szoftverfejlesztés hagyományos megközelítését alkalmazzák. Ebben a paradigmában az alkalmazásokat és rendszereket eljárások, függvények és utasítások segítségével írják le. Az imperatív paradigmák közé tartozik a strukturált programozás és az objektumorientált programozás (OOP).

Deklaratív paradigmák: A deklaratív paradigmák a szoftverfejlesztés egy újabb megközelítését képviselik. Ebben a paradigmában az alkalmazásokat és rendszereket leíró kódot olyan nyelven írják, amely inkább kijelentések, mint utasítások formájában írja le a problémát. A deklaratív paradigmák közé tartozik a funkcionális programozás.

Logikai paradigmák: A logikai paradigmák a szoftverfejlesztés olyan megközelítése, amelyben az alkalmazásokat és rendszereket logikai kifejezéseken alapuló szabályrendszerrel írják le. A logikai paradigmák közé tartozik a logikai programozás és a szabály alapú programozás.

Komponens alapú paradigmák: A komponens alapú paradigmák a szoftverfejlesztés egy modern megközelítése, amelyben az alkalmazásokat és rendszereket előre megírt és tesztelt összetevőkből állítják össze, amelyeket a fejlesztők újra felhasználhatnak. A komponens alapú paradigmákhoz tartozik a szolgáltatás-orientált architektúra (SOA) és a mikroszolgáltatások.

A szoftverfejlesztési paradigmák alapvetően meghatározzák a szoftverfejlesztési folyamatot, és meghatározzák a fejlesztői csapatok szerepét és az alkalmazások architektúráját.

A különböző paradigmák közötti választás a szoftverfejlesztési célkitűzéseknek, a projektkörnyezetnek és a szervezeti kultúrának megfelelően változó lehet. A megfelelő paradigmák kiválasztása az alkalmazás és rendszer hatékonyabb tervezéséhez és fejlesztéséhez vezethet.

5.7.3.1 Extreme Programming

Az Extreme Programming (XP) egy agilis szoftvertervezési módszertan, amely a szoftverfejlesztés során a folyamatos tesztelés, tervezés és kommunikáció fontosságára helyezi a hangsúlyt.

Az XP célja, hogy rugalmasabbá tegye a szoftverfejlesztést, és lehetővé tegye a gyors és hatékony választ az ügyfelek igényeire és az esetleges változásokra.

Az XP főbb elemei a **rövid iterációk**, a **feladatok páros programozása**, az egyszerű tervezés, a **folyamatos integráció** és a **folyamatos tesztelés**.

5.7.4 Szoftverfejlesztési Platform

A szoftverfejlesztési platform olyan hardver- és szoftverinfrastruktúrát jelent, amely lehetővé teszi a szoftveralkalmazások fejlesztését, tesztelését és futtatását. A szoftverfejlesztési platformok lehetnek helyi (On-Premise) vagy felhő alapúak, és tartalmazhatnak többek között fejlesztői eszközöket, adatbázis-kezelő rendszereket, alkalmazásintegrációs eszközöket, tesztelési eszközöket és verziókezelő rendszereket.

A helyesen kiválasztott és hatékonyan használt platformok lehetővé teszik a fejlesztők számára, hogy gyorsabban és hatékonyabban fejlesszenek, teszteljenek és telepítsenek alkalmazásokat.

Java Platform, Enterprise Edition (Java EE): A Java EE egy olyan fejlesztői platform, amelyet nagyvállalati szoftveralkalmazások fejlesztéséhez használnak. Tartalmazza a Java programozási nyelvet, az Oracle GlassFish alkalmazásszervert, az Eclipse IDE-t és más szükséges komponenseket.

Microsoft .NET Framework: Az .NET Framework egy fejlesztői platform, amelyet a Microsoft fejlesztett ki a Windows operációs rendszerhez. Tartalmazza a .NET programozási nyelvet, az Visual Studio IDE-t és az IIS webkiszolgáló-t.

Android fejlesztői platform: Az Android fejlesztői platform az Android mobilalkalmazások fejlesztéséhez használt szoftver- és hardverkörnyezet

iOS fejlesztői platform: Az iOS fejlesztői platform az Apple iOS operációs rendszeréhez készült mobilalkalmazások fejlesztéséhez használt szoftver- és hardverkörnyezet. Tartalmazza az Xcode IDE-t, az iOS SDK-t és a tesztelési eszközöket.

Node.js: A Node.js egy nyílt forráskódú, cross-platform JavaScript runtime környezet, amely lehetővé teszi a fejlesztők számára a szerveroldali JavaScript fejlesztését.

Ruby on Rails: A Ruby on Rails egy webes alkalmazásfejlesztő keretrendszer, amely a Ruby programozási nyelvhez kapcsolódik. A keretrendszer célja a gyors és hatékony webalkalmazás-fejlesztés támogatása.

React Native: A React Native egy keretrendszer, amely lehetővé teszi a mobilalkalmazások fejlesztését a React JavaScript keretrendszerrel. A React Native egyre népszerűbbé válik a cross-platform mobilalkalmazás-fejlesztés területén.

Laravel: A Laravel egy PHP alapú webalkalmazás-fejlesztő keretrendszer, amely a gyors és hatékony webalkalmazás-fejlesztést támogatja.

5.7.5 Implementáció - általános folyamat

Az implementáció általános folyamatának lépései

Követelmények azonosítása: Az első lépés a követelmények azonosítása, amelyek az alkalmazás számára szükségesek. Ez magában foglalja az üzleti és felhasználói követelményeket, a rendszerspecifikációkat és az eszközök és platformok követelményeit.

Tervezés: Ebben a lépésben a fejlesztők terveket készítenek az alkalmazásra vonatkozóan. Ez magában foglalja az alkalmazás architektúráját, a szükséges funkciókat, az adatmodellt és az alkalmazás felépítését.

Fejlesztés: A tervezés után a fejlesztők elkezdik az alkalmazás kódolását a tervek alapján. A fejlesztőknek alkalmazkodniuk kell a fejlesztési környezetekhez és az alkalmazás specifikációihoz.

Tesztelés: Az alkalmazás tesztelése szükséges az alkalmazás hibáinak azonosításához és javításához. A tesztelés során a fejlesztők különböző teszteseteket hajtanak végre, hogy ellenőrizzék az alkalmazás működését, és biztosak legyenek abban, hogy az alkalmazás megfelel a követelményeknek.

Telepítés: Az alkalmazás telepítése a végfelhasználók rendszereire történik. Ez magában foglalhatja az alkalmazás konfigurálását, az adatbázis és más összetevők telepítését, valamint az alkalmazás beállításainak finomhangolását.

Az implementáció folyamata változó lehet az alkalmazás méretétől, a fejlesztők számától, az alkalmazás környezetétől és az üzemeltetési igényektől függően.

5.7.6 Szoftver implementáció - általános folyamat

A szoftver implementáció általános folyamatának lépései

Tesztelés és minőségellenőrzés: A fejlesztők és a tesztelők tesztelik a szoftvert, és ellenőrzik, hogy az megfelel-e az előírt minőségi követelményeknek.

Csomagolás (Packaing): A szoftvert csomagolják, hogy az könnyen telepíthető legyen a felhasználói rendszerekbe.

Telepítés: A szoftvert telepítik a célhardverekre vagy az ügyfél szervereire.

Integráció: A szoftvert integrálják más rendszerekkel, például adatbázisokkal vagy alkalmazásokkal.

Konfiguráció: A szoftvert beállítják az ügyfél igényei szerint, például a felhasználói jogosultságok és a biztonsági beállítások tekintetében.

Migráció: Az adatokat az előző rendszerekből a szoftverbe importálják, és az új rendszerből adatokat exportálnak más rendszerekbe.

Tesztelés és validálás: A szoftvert tesztelik és validálják a végfelhasználói igényeknek megfelelően.

Üzembe helyezés: Az üzembe helyezés során a szoftvert már használják a valós környezetben.

Karbantartás és frissítések: A szoftvert karbantartják és frissítik az életciklusa során. Ez magában foglalja a hibajavításokat, a biztonsági frissítéseket és az új funkciók bevezetését.

Fontos megjegyezni, hogy az implementáció folyamata nagymértékben függ az adott szoftverfejlesztési módszertantól és az alkalmazott technológiáktól.

5.7.7 Implementáció hátrányai

Az implementáció hátrányai

Nagy idő- és erőforrás-igény: Az implementáció időigényes folyamat, ami sok erőforrást igényel a szervezet számára, beleértve az anyagiakat, az embereket és az infrastruktúrát.

Kockázatok: Az implementáció során számos kockázat merülhet fel, amelyek befolyásolhatják a projekt sikerét, például a technikai hibák, a biztonsági problémák vagy az alkalmazottak ellenállása.

Megfelelőségi problémák: Az implementáció során a szervezetnek betartania kell a jogszabályokat és az iparági előírásokat. Ha nem sikerül teljesíteni ezeket a követelményeket, akkor komoly jogi következményekkel kell szembenézni.

Átállási nehézségek: Az új rendszerek és folyamatok bevezetése nehézségeket okozhat a munkatársak számára, akiknek új képességekre és rutinokra van szükségük.

Alkalmazott ellenállás: Az alkalmazottak gyakran ellenállnak a változásoknak, és nehéz lehet őket meggyőzni arról, hogy az új rendszerek vagy folyamatok előnyösek számukra vagy a szervezet számára.

5.7.8 Fejlesztői Platform

A fejlesztői platform egy olyan szoftver- és hardverkörnyezet, amelyet a szoftverfejlesztők használnak az alkalmazások fejlesztésére, tesztelésére és üzembe helyezésére. A fejlesztői platform általában tartalmazza az alkalmazások fejlesztéséhez szükséges összes eszközt, köztük a programozási nyelv(ek) fordítóprogramjait, fejlesztői környezetet (IDE), tesztelő eszközöket, adatbázis-szervereket, web-szervereket és más komponenseket.

A fejlesztői platformok lehetnek különböző típusúak, például asztali (desktop) alkalmazásokhoz, mobil alkalmazásokhoz vagy webes alkalmazásokhoz használhatók. A különböző platformok különböző eszközöket és technológiákat használnak, és a választás függ a fejlesztési célkitűzésektől és az alkalmazás követelményeitől.

A fejlesztői platformok lehetnek ingyenes vagy fizetős licenszűek, és lehetnek nyílt forráskódúak vagy zárt forráskódúak. Sok fejlesztői platform támogatja a több platformra történő fejlesztést, amely lehetővé teszi az alkalmazások könnyebb portolását és a szélesebb körű elérhetőséget. A megfelelő fejlesztői platform kiválasztása fontos szerepet játszik az alkalmazások minőségének és hatékonyságának javításában, valamint az üzleti célok elérésében.

5.7.9 Fejlesztőeszköz (IDE)

Az IDE (**Integrated Development Environment**) egy olyan fejlesztői platform, amely integrálja az összes eszközt és funkciót, amelyek szükségesek egy adott programozási nyelvben történő fejlesztéshez.

Korszerű Integrated Development Environment (IDE) tulajdonságok

Kódszerkesztő funkciók: az IDE rendelkezik szintaxis kiemeléssel, automatikus kódkiegészítéssel, hibajavítással és más hasznos kódszerkesztő funkciókkal.

Integrált hibakereső és profilozó eszközök: lehetővé teszik a hibák és a teljesítményproblémák gyors azonosítását. Az IDE tartalmazza a hibakeresést és a hibadokumentálást segítő funkciókat.

Integrált verziókezelő: az IDE támogatja a verziókezelést, lehetővé téve a kódváltozások nyomon követését és az együttműködést a fejlesztők között.

Támogatás különböző programozási nyelvekhez: az IDE támogatja a különböző programozási nyelveket és platformokat, amelyek lehetővé teszik a keresztplatformos fejlesztést.

Egyszerű használat: az IDE könnyen használható, gyorsan betanulható felhasználói felülettel rendelkezik, és segít a fejlesztőknek hatékonyabban dolgozni.

Multi-platform támogatás: Az IDE-t minden platformon (Windows, Linux, Mac stb.) használni lehet.

Nyelvi támogatás: Az IDE támogatja a különböző programozási nyelveket, például Java, Python, C++, stb.

Integráció: Az IDE lehetőséget ad a különböző fejlesztői eszközök, például a verziókezelő rendszerek (pl. Git), az automatizált tesztelő rendszerek, vagy az adatbázisok integrálására.

Plug-inok: Az IDE lehetőséget ad a különböző plug-in-ok telepítésére és használatára, amelyek bővíthetik az IDE funkcióit.

Felhőintegráció: Az IDE képes az alkalmazások fejlesztésének támogatására a felhőben, például a fejlesztői környezet és a tesztelési eszközök kialakításával.

Gyorsaság: Az IDE gyors és hatékony működésével segíti a fejlesztőket a hatékony munkában és a kód minőségének javításában.

5.7.10 Szoftver implementáció - eszközök

A szoftver implementáció eszközei különböző programozási nyelveket, fejlesztői környezeteket, verziókezelő eszközöket, tesztelő és hibakereső eszközöket, valamint build automatizáló eszközöket tartalmaznak. Az alábbiakban felsorolunk néhány példát ezek közül:

- **Programozási nyelvek**: például Java, Python, C#, JavaScript, PHP stb.
- **Fejlesztői környezetek**: például Eclipse, Visual Studio, IntelliJ stb.
- **Verziókezelő eszközök**: például Git, SVN stb.
- **Tesztelő és hibakereső eszközök**: például JUnit, Selenium, NUnit, IntelliJ Debugger stb.
- **Build automatizáló eszközök**: például Maven, Gradle stb.

Ezen eszközök használata segíti a szoftverfejlesztőket a hatékonyabb és gyorsabb fejlesztésben, valamint a kód minőségének javításában.

5.7.11 Programnyelvek

- C : **Imperatív Modell**
- Python : **Deklaratív Modell**
- C++, Java, C# : **Objektum Orientált Modell**

5.7.11.1 Szoftverfejlesztő nyelvek fő jellegzetességei

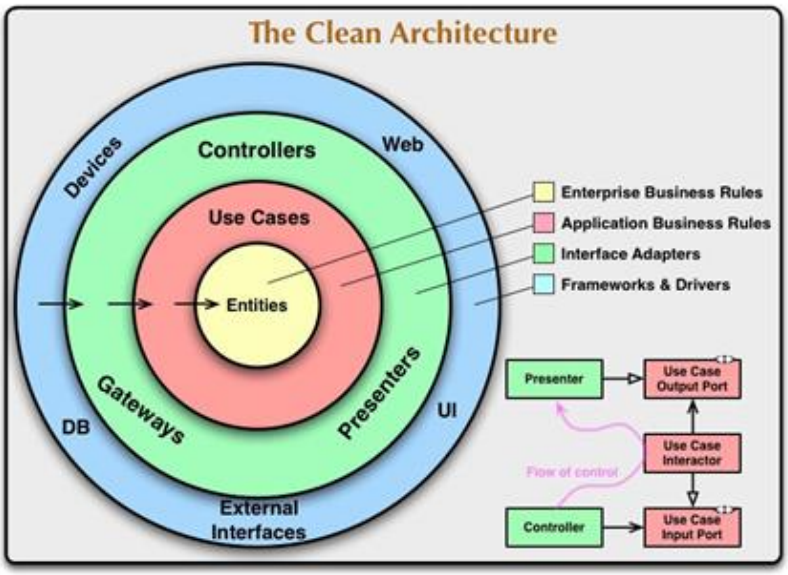
- Egy Funkcionális programozási nyelvben az utasítások sorrendje tetszőleges.
- Egy Objektum Orientált nyelv nem alkalmas párhuzamosításra.
- Egy Imperatív programozási nyelv mutálhatja az adatait.

5.7.11.2 Clean Code direktíva

A Clean Code egy könyv, amelyet **Robert C. Martin** írt a szoftvertervezés és -fejlesztés témakörében. A könyv fő üzenete az, hogy a jó minőségű kód írása és karbantartása kulcsfontosságú a hatékony és eredményes szoftvertervezéshez és -fejlesztéshez.

A Clean Code a szoftvertervezési elvek és gyakorlatok gyűjteménye, amelyek segítenek a fejlesztőknek a hatékony és karbantartható kód írásában. A Clean Code az olvashatóság, az egyszerűség, a modularitás és a tesztelhetőség szempontjait helyezi előtérbe a szoftverfejlesztés során.

A Clean Code-nak számos előnye van, mint például a karbantarthatóság növelése, a hibakeresés egyszerűsítése, a hatékonyság növelése és a kód minőségének javítása általánosságban.



5.7.12 Alkalmazás tervezés szempontjai

1. **Rendszer elosztás**: Az alkalmazás tervezésekor figyelembe kell venni, hogy az alkalmazás hol fog futni, milyen környezetben, milyen architektúrában. Az alkalmazásnak lehetnek olyan részei, amelyek különböző rendszereken, szervereken futnak, így ezeknek az egymással való kommunikációja is meg kell tervezni.

2. **Modul struktúra**: Az alkalmazásnak általában több modulból áll, amelyek egymással kommunikálnak. Fontos, hogy a modulok struktúrája, egymáshoz való kapcsolódása jól legyen megtervezve, hogy az alkalmazás könnyen karbantartható és bővíthető legyen.

3. **Vezérlési szerkezet**: Az alkalmazás vezérlési szerkezete határozza meg az adatok áramlását és feldolgozását az alkalmazásban. Fontos, hogy ez a szerkezet jól átgondolt legyen, hogy az alkalmazás hatékonyan működjön.

4. **Értékelési módszertan**: Az alkalmazás tervezésének során fontos, hogy a fejlesztők folyamatosan értékeljék az elkészült modulokat és az egész alkalmazást is. Ehhez számos értékelési módszertan létezik, amelyek közül célszerű az adott projektnek és fejlesztői csapatnak megfelelőt választani.

5. **Dokumentációs struktúra**: Az alkalmazás tervezése során fontos, hogy az elkészült munkákat dokumentáljuk. Az alkalmazás tervezésének dokumentációja segíti a fejlesztői csapat kommunikációját, illetve más fejlesztőknek is megkönnyíti az alkalmazás megértését és karbantartását.

5.7.13 Adatbázis fejlesztési alapelvek

- **BCNF**: A Boyce-Codd normál forma, mely egy adatbázis tervezési elv, amely azt mondja ki, hogy egy relációs adatbázis akkor van jól normalizálva, ha minden nem-kulcs attribútum funkcionálisan függ a teljes kulcsrészektől.

- **3NF**: A harmadik normál forma, amely egy másik adatbázis tervezési elv, amely azt mondja ki, hogy egy relációs adatbázis akkor van jól normalizálva, ha minden nem-kulcs attribútum nem tranzitív funkcionális függőségben áll a kulcsrészektől.

- **SROT**: A "Single Source of Truth" (SROT) az a megközelítés, amely az adatok kezelésére és az információk terjesztésére vonatkozik. A SROT azt jelenti, hogy az adatoknak egyetlen forrása van, ami az egész szervezet számára hivatalos és megbízható adatforrás. Ez azt jelenti, hogy az adatokat csak egyszer kell rögzíteni, és mindenki ugyanazon adatokat látja, amikor arra szüksége van.

- **ACID**: A tranzakciókezelés az adatbázis-kezelő rendszer (DBMS) által nyújtott szolgáltatás, amely lehetővé teszi az ügyfelek (felhasználók) számára az adatbázis-kezelő rendszer tranzakcióinak végrehajtását. A tranzakció egy műveletsorozat, amely általában adatbázis-műveleteket tartalmaz, például adatbeszúrást, adatmódosítást és adattörlést. A tranzakciókezelés fontos szerepet játszik az adatbázisok biztonságos és hatékony kezelésében, mivel biztosítja, hogy az adatbázis műveletek végrehajtása összhangban van a tervezett üzleti folyamatokkal, és a végrehajtás során az adatok épsége megörződik.

  A tranzakciókezelésnek négy fő ACID tulajdonsága

  - **Elemiség** (Atomicity): A tranzakció egyszerre hajtja végre vagy semmit, vagyis az összes művelet sikeres végrehajtásakor az eredményeket véglegesen rögzítik az adatbázisban, ellenkező esetben minden változtatást visszaállítanak a tranzakció megkezdése előtti állapotra.

  - **Következetesség** (Consistency): A tranzakciók végrehajtása során az adatbázis minden állapota érvényes és összhangban van az előírt szabályokkal, így az adatbázis állapota mindig érvényes az üzleti logikára nézve.

- **Izoláció** (Isolation): A tranzakciók külön-külön futnak, mintha egymástól függetlenek lennének, még akkor is, ha ténylegesen egymástól függenek. Ez azt jelenti, hogy a tranzakciók eredményei nem zavarják egymást, és az adatbázis kezelő rendszer biztosítja az egymástól független végrehajtást.

- **Tartósság** (Durability): Az adatbázis-kezelő rendszer biztosítja, hogy az adatbázis tartalmazza az összes tranzakció sikeres végrehajtásából származó eredményeket, még akkor is, ha az adatbázis-kezelő rendszer vagy a számítógép rendszer hiba miatt meghibásodik.

- **MDM**: Master Data Management egy adatkezelési stratégia, amely azon törekszik, hogy egységes nézetet biztosítson egy vállalat valamennyi adatáról, különösen a legfontosabb, vagy "mester" adatokról. A mester adatok olyan adatok, amelyeket a vállalat számos alkalmazása használ, például ügyfélnevek vagy termékazonosítók. Az MDM célja az, hogy biztosítsa az adatok konzisztenciáját és pontosságát az összes rendszerben és alkalmazásban, amelyek használják ezeket az adatokat.

5.7.14 Szoftvertervezési hibagyűjtemény

1. **Scoping Woes.**
2. **Not Casting Your Net Widely.**
3. **Just Focusing on Functions.**
4. **Box and Line Descriptions.**
5. **Forgetting That It Needs to be Built.**
6. **Lack of Platform Precision.**
7. **Making Performance and Scalability Assumptions.**
8. **DIY Security.**
9. **No Disaster Recovery.**
10. **No Backout Plan.**

1. **Hatókör meghatározás nehézségek**: Az alkalmazás hatáskörének meghatározása nehézséget okozhat a tervezési folyamat során.
2. **Nem megfelelő adatkör**: Ha az alkalmazás adatköre túl szűk, akkor az nem tudja kielégíteni az üzleti igényeket. Ha túl széles, akkor az alkalmazás bonyolulttá és kezelhetetlenné válhat.
3. **Funkciókra összpontosítás**: Elvezethet egy olyan rendszerhez, amely nem nyújt elég hatékony megoldást az üzleti problémára.
4. **Egyszerű blokkdiagramok használata**: nem biztosít elég részletességet az alkalmazás tervezéséhez.
5. **A kivitelezhetőség háttérbe szorítása**: Az alkalmazás tervezésekor fontos, hogy az kivitelezhető legyen, és megfelelő erőforrások álljanak rendelkezésre a megvalósításhoz.
6. **Hiányos platform specifikációk**: A tervezés során meg kell határozni a platform specifikációkat, hogy az alkalmazás megfelelően működjön.
7. Túl optimista teljesítmény és skálázhatósági előrejelzések: nem biztos, hogy megfelelően működni fog a valóságban.
8. **Helyi biztonsági megoldások alkalmazása**: A helyi biztonsági megoldások alkalmazása a szoftver tervezésekor kockázatot jelenthet, ha azok nem megfelelően védik az alkalmazást a külső támadásoktól.
9. **Hiányzó katasztrófa-előrejelzés**: Az alkalmazás tervezésekor nem szabad elfelejteni a katasztrófa-előrejelzést, vagyis azt, hogy mi történik, ha valami nem működik a rendszerben.
10. **Hiányzó visszaállítási terv**: Az alkalmazás tervezésekor szükséges egy visszaállítási terv kidolgozása, hogy ha valami hiba történik, akkor az alkalmazás működése minél hamarabb helyreálljon.

5.8 Rendszer leírás

5.8.1 A Rendszerterv

A Rendszerterv célja a tervezett rendszer részletes leírása és specifikálása. Létrehozása általában a fejlesztési folyamat egy korai szakaszában történik, amikor a rendszer általános követelményeit már meghatározták, és az alapos tervezés kezdeti szakaszában vannak. Általános célja, hogy meghatározza a rendszer pontos specifikációit, és biztosítsa a tervezési, fejlesztési és telepítési folyamatok hatékony végrehajtását.

5.8.2 A Rendszerterv dokumentum

Olyan részletes leírása egy rendszernek, amely magában foglalja

- **a rendszer célját**
- **felépítését**
- **funkcióit**
- **követelményeit**
- **tervezési döntéseit**
- **az implementáció és tesztelés részleteit**

5.8.3 A Rendszerterv dokumentum témakörök

Bevezető

Ez a fejezet általában a dokumentum célját, a rendszer áttekintését, valamint a dokumentum felépítését mutatja be.

Rendszerleírás

Ez a fejezet általában az üzleti cél leírását, a rendszer felépítését, a komponensek és a funkciók leírását tartalmazza.

Rendszerkövetelmények

Ez a fejezet általában a rendszer követelményeit tartalmazza, például a funkcionális és nem funkcionális követelményeket, a teljesítménykövetelményeket, az interfész követelményeket és a biztonsági követelményeket.

Rendszertervezés

Ez a fejezet általában a rendszer architektúráját, a hardver és szoftver összetevők tervezését, az adatmodellt és az adatbázist tervezését, valamint az interfészek tervezését mutatja be.

Implementáció és tesztelés

Ez a fejezet általában a rendszer implementációjának és tesztelésének részleteit tartalmazza, beleértve az integrációs teszteket és a rendszerminőség-ellenőrzést.

Projektmenedzsment

Ez a fejezet általában a projekt tervezését, irányítását és ellenőrzését, valamint a projekt szereplőinek és a projektmenedzsment folyamatának leírását tartalmazza.

Dokumentumlista

Ez a fejezet általában a dokumentum többi részére való hivatkozást és a dokumentumváltozatok kezelését tartalmazza.

5.8.4 A Rendszerterv dokumentum felépítés

1. **Bevezetés**: A dokumentum célja, áttekintése és a szükséges definíciók meghatározása.
2. **Projekt áttekintése**: A projekt célja, a projekt szereplőinek listája, a követelmények áttekintése.
3. **Rendszerterv architektúra**: A rendszer architektúrájának áttekintése, a rendszer blokkdiagramja, a komponensek és az interfészek leírása.
4. **Rendszerspecifikáció**: A rendszer általános specifikációja, a szolgáltatások és funkciók leírása, az üzleti folyamatok leírása, a felhasználói követelmények leírása, az adatmodell és a felhasználói felület specifikációja.
5. **Fejlesztési környezet**: A fejlesztési környezet, a tesztelési környezet és az élesítési környezet leírása.
6. **Projektterv**: A projekt menetrendje, a szükséges erőforrások, az ütemezés és a projekt költségvetése.
7. **Projektmenedzsment**: A projektvezetési módszerek, a kommunikációs terv, a kockázatok kezelése és az eszközök leírása.
8. **Változáskezelés**: A változások kezelése, a változások követése és dokumentálása, a változások hatásának értékelése.
9. **Karbantartás és támogatás**: A rendszer karbantartási és támogatási igényeinek leírása, a karbantartási és támogatási folyamatok leírása.
10. **Rendszerbiztonság**: A rendszer biztonsági funkcióinak és követelményeinek leírása, a biztonsági protokollok és a védelmi mechanizmusok leírása.

A pontos tartalom az adott projekttől és az ügyfél igényeitől is függhet.

6 Megvalósítási minta készlet

6.1 Szoftver Architektúra Nézőpont (Viewpoint)

A szoftverarchitektúra nézőpont egy olyan módszer, amely lehetővé teszi a szoftverarchitektúra különböző nézőpontjainak azonosítását és leírását a tervezési folyamat során. Olyan képességeket biztosít, amelyekkel az architektek a szoftverarchitektúra különböző szempontjait, például a **funkcionalitást**, a **teljesítményt**, a **biztonságot** és az **adatokat**, egységes és koherens módon képesek leírni.

A szoftverarchitektúra viewpoint négy fő elemből áll:

- nézőpont
- elemek
- kapcsolatok
- a séma leírása

A **nézőpont** azonosítja az architektúra célját és a szempontot, amelynek alapján az architektúrát leírják.

Az **elemek** az architektúra elemeit azonosítják, például az alkalmazás komponenseit és a szolgáltatásokat.

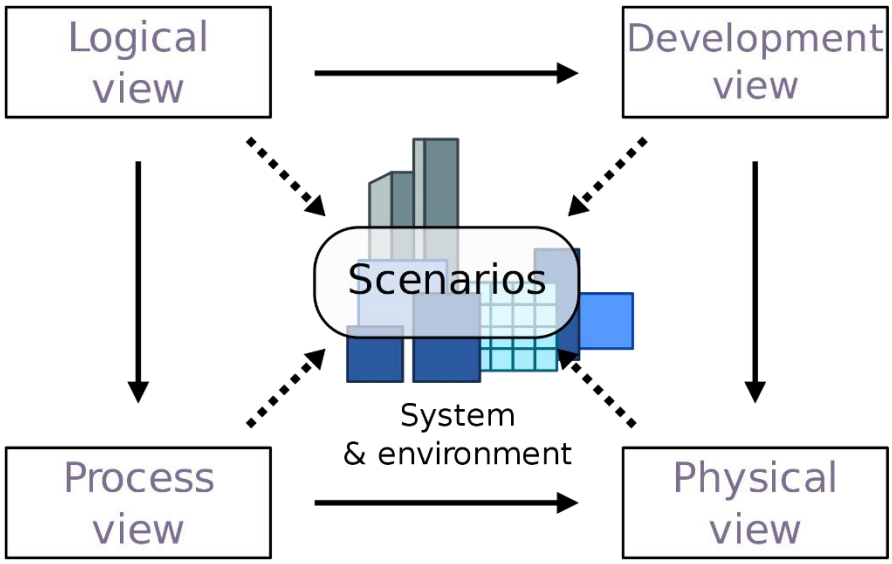
A **kapcsolatok** az elemek közötti kapcsolatokat írják le, például az adatfolyamokat és a szolgáltatások közötti kommunikációt.

A **séma** leírásának célja, hogy megmutassa, hogy az elemek és kapcsolatok hogyan illeszkednek egymáshoz a nézőpont céljának megfelelően.

Az egyes viewpoint-ok által biztosított információk segítenek a fejlesztőknek és a tervezőknek az architektúra tervezésében és megértésében azzal, hogy lehetővé teszi, hogy az architektúra különböző szempontjait külön-külön és koherens módon kezeljék, ami segít a fejlesztőknek az architektúra hatékony és kohéziós kialakításában.

A szoftverarchitektúra **leírásait** általában **nézetekbe szervezik**, amelyek analógok az építészetben készített különböző típusú **tervrajzokkal**.

Minden egyes nézet a rendszerrel kapcsolatos kontextus egy csoportjával foglalkozik, a nézet konvencióit követve, ahol **a nézet egy olyan specifikáció, amely leírja az olyan nézetben használandó jelöléseket, modellezési és elemzési technikákat**, amely a szóban forgó architektúrát az érdekeltek egy adott csoportjának és az ő érdeklődésüknek a szemszögéből fejezi ki (ISO/IEC/IEEE 42010). A nézőpont nem csak a megfogalmazott (azaz a kezelendő) kontextust határozza meg, hanem a megjelenítést, a használt modelltípusokat, az alkalmazott konvenciókat és a konzisztencia (megfeleltetési) szabályokat is, hogy a nézőpont konzisztens maradjon más nézőpontokkal.



Nézőpontok

Funkcionális nézőpont: Leírja az alkalmazás funkcionális jellemzőit és követelményeit.

Logikai nézőpont: Leírja az alkalmazás logikai felépítését, mint például a rétegzett architektúrát vagy a moduláris felépítést.

Adat nézőpont: Leírja az alkalmazás adatmodelljét, az adatok tárolásának és kezelésének módját.

Teljesítmény nézőpont: Leírja az alkalmazás teljesítményének javítását célzó tervezési döntéseket, például a gyorsítótárak vagy a többszálú feldolgozás használatát.

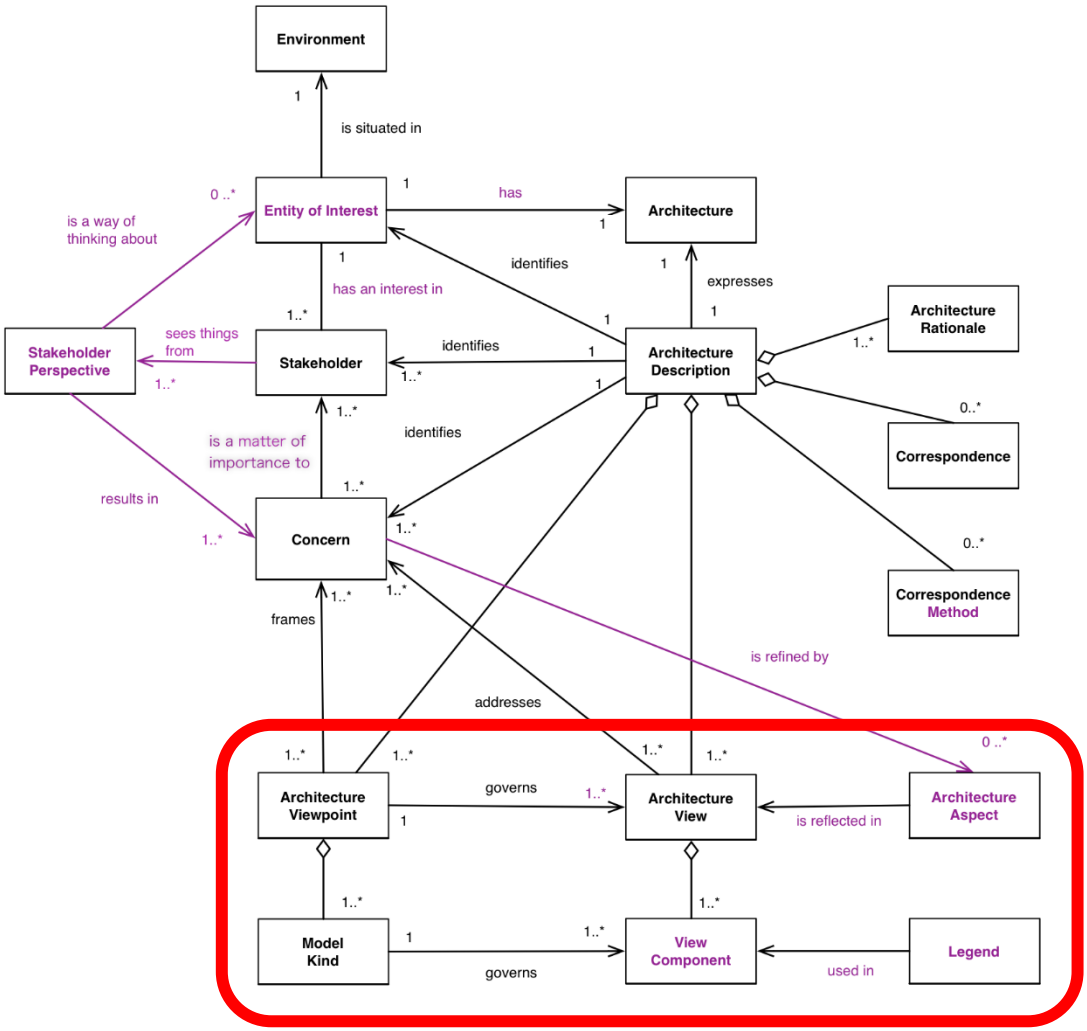
Biztonsági nézőpont: Leírja az alkalmazás biztonsági követelményeit és azokat az architekturális döntéseket, amelyek biztosítják ezeket a követelményeket.

Rendszerüzemeltetési nézőpont: Leírja az alkalmazás üzemeltetésének és karbantartásának folyamatát, valamint az alkalmazás felügyeletét és diagnosztizálását célzó technikákat.

Implementációs nézőpont: Leírja az alkalmazás megvalósítását, beleértve a programozási nyelvet, az architekturális mintázatokat és a fejlesztői eszközöket.

Ezek a nézőpontok segíthetnek a tervezőnek a szoftverarchitektúra hatékonyabb és koherensebb tervezésében és megértésében. Az egyes nézőpontok olyan területekre összpontosítanak, amelyek a szoftverarchitektúra különböző szempontjait jelentik, és lehetővé teszik a fejlesztőknek, hogy az architektúra különböző szempontjait külön-külön és koherens módon kezeljék.

ISO/IEC/IEEE 42010



6.1.1 Szoftver Architektúra - Logikai nézőpont

A Szoftver Architektúra Logikai nézőpontjában a fókusz a rendszer logikájára és struktúrájára irányul.

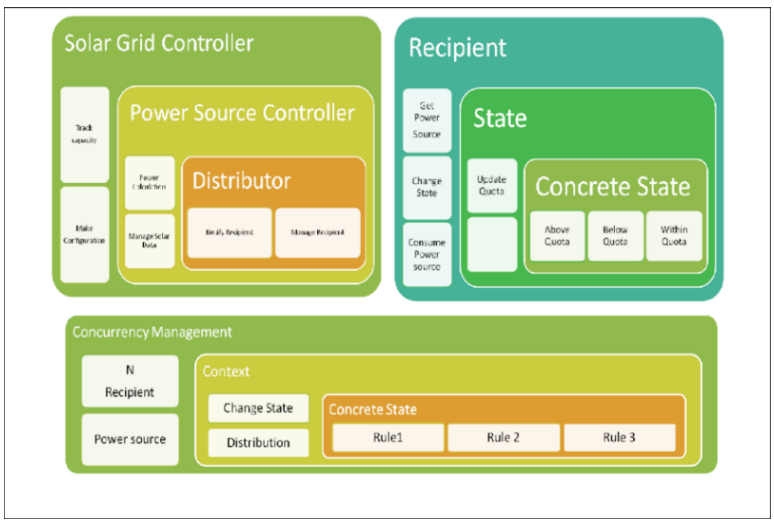
UML modellalkotó eszközök: Az UML modellalkotó eszközök, mint például az Enterprise Architect, a Visual Paradigm vagy a MagicDraw, segítenek a logikai nézőpont tervezésében, mivel lehetővé teszik az UML modellek létrehozását és szerkesztését. Ez az eszközök segítségével lehetőség van a rendszer különböző osztályainak, interfészeinek, metódusainak és más logikai elemeknek a leírására.

Programozási nyelvek: A programozási nyelvek, mint például a Java vagy a C#, fontos szerepet játszanak a logikai nézőpont tervezésében. Ezeknek a nyelveknek a használata lehetővé teszi a rendszer különböző logikai elemekkel való létrehozását és megvalósítását.

Szövegszerkesztők: A szövegszerkesztők, mint például a Notepad++, a Visual Studio Code vagy az Eclipse, szintén fontos szerepet játszanak a logikai nézőpont tervezésében. Ezekkel az eszközökkel lehetőség van a kódolásra, tesztelésre és hibakeresésre, ami a logikai nézőpont hatékonyabb tervezését teszi lehetővé.

Tesztelő eszközök: A tesztelő eszközök, mint például a JUnit vagy a NUnit, lehetővé teszik a logikai nézőpont alapos tesztelését és hibakeresését. Ezekkel az eszközökkel lehetőség van a kód lefedettségének ellenőrzésére, valamint a kód hibáinak azonosítására és javítására.

Ezek az eszközök lehetővé teszik a hatékonyabb logikai nézőpont tervezését és megvalósítását, ami elengedhetetlen az alkalmazások hatékony működéséhez és fejlesztéséhez. **Az eszközök kiválasztása az adott projekthez és az alkalmazás igényeihez igazítva segíti a hatékonyabb szoftvertervezést és -fejlesztést**.



6.1.2 Szoftver Architektúra - Funkcionális nézőpont

A szoftverarchitektúra funkcionális nézőpontjának támogatására számos eszköz és technika áll rendelkezésre, amelyek segítenek az alkalmazás funkcionalitásának leírásában és dokumentálásában.

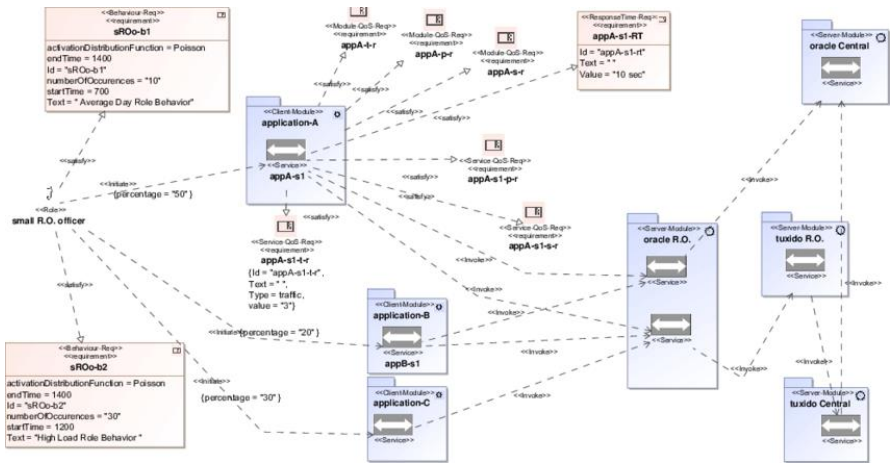
Use case diagramok: A Use case diagramok segítenek az alkalmazás funkcionalitásának leírásában a felhasználók és a rendszer közötti interakciók szempontjából. Ezek a diagramok a rendszer által nyújtott funkciókat, a felhasználók által végzett műveleteket és az alkalmazás céljait és használati eseteit írják le.

Funkcionális specifikációk: A funkcionális specifikációk olyan dokumentumok, amelyek leírják az alkalmazás funkcionalitását a felhasználói szempontból. Ez magában foglalhatja az alkalmazás célját, a felhasználók számára elérhető funkciókat, a műveletek és a felhasználói interakciók részletes leírását.

Szerződéses alapú tervezés: A szerződéses alapú tervezés egy módszer, amely az alkalmazás részeire vonatkozó meghatározott szerződéseket használ az alkalmazás megfelelő működésének biztosításához. Ez a megközelítés lehetővé teszi az alkalmazás részeinek egyszerű tesztelését és karbantartását.

Prototípus készítése: A prototípus készítése lehetővé teszi az alkalmazás funkcionalitásának tesztelését és a visszajelzés alapján az alkalmazás tervezésének finomítását. Ez lehetővé teszi az alkalmazás problémáinak korai azonosítását és megoldását, mielőtt az alkalmazás fejlesztése befejeződne.

Ezek az eszközök és technikák segíthetnek az architektusoknak a szoftverarchitektúra funkcionális nézőpontjának hatékonyabb kezelésében és dokumentálásában.



6.1.3 Szoftver Architektúra - Információ nézőpont

A szoftver architektúra adat nézőpontjának támogatására számos eszköz és technika áll rendelkezésre, amelyek segítenek az adatok leírásában és dokumentálásában.

Adatmodellező eszközök: Az adatmodellező eszközök lehetővé teszik az adatokat leíró modell kialakítását és dokumentálását. Ezek az eszközök segítenek az adatok struktúrájának és kapcsolatainak leírásában, és segítik az architektusokat az adatok és a folyamatok közötti kapcsolatok azonosításában.

Adatfolyam-diagramok: Az adatfolyam-diagramok a folyamatok és az adatok közötti kapcsolatokat írják le. Segítenek az adatok útvonalának és az adatok tárolásának nyomon követésében.

Adatfolyam elemzés: Az adatfolyam-elemzés egy olyan technika, amely az adatok áramlását és a folyamatok közötti adatokat kezeli. Az adatfolyam-elemzés segíthet az adatok hatékonyabb használatában, az adatok áramlásának optimalizálásában és a rendszer hatékonyságának növelésében.

Adatbiztonsági eszközök: Az adatbiztonsági eszközök biztosítják az adatok védelmét az illetéktelen hozzáféréstől, manipulációtól és más fenyegetésektől. Ez magában foglalja az adatvédelmi eszközöket, az adatvédelmi politikákat, a hozzáférés-kezelő rendszereket és más biztonsági megoldásokat.

6.1.4 Szoftver Architektúra - Implementációs nézőpont

A Szoftver Architektúra Implementációs nézőpontjában a fókusz a rendszer megvalósítására és implementálására irányul.

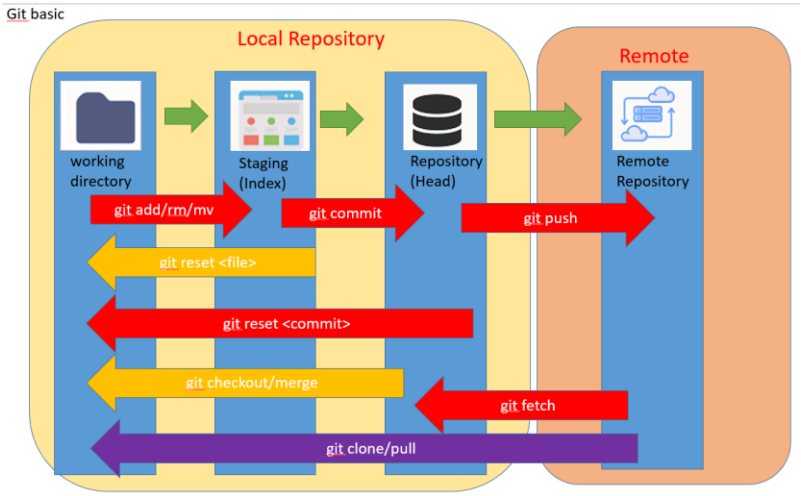
Integrált fejlesztői környezetek (IDE): Az IDE-k, mint például az Eclipse, a Visual Studio vagy a IntelliJ IDEA, lehetővé teszik a kódírás és a hibakeresés hatékonyabb megvalósítását. Ezek az eszközök általában tartalmaznak integrált fordítókat, tesztelő eszközöket és verziókezelő rendszereket, amelyek lehetővé teszik a kód hatékonyabb kezelését és tesztelését.

Verziókezelő rendszerek: A verziókezelő rendszerek, mint például a Git, a Subversion vagy a Mercurial, lehetővé teszik a kódváltozatok nyomon követését, a változatok közötti különbségek kezelését és az együttműködést a különböző fejlesztők között. Ez elengedhetetlen a hatékonyabb kódkezeléshez és a verziók közötti visszatéréshez.

Build eszközök: A build eszközök, mint például az Apache Maven vagy az Apache Ant, lehetővé teszik a szoftver build folyamatának automatizálását, az alkalmazás függőségeinek kezelését és a kódminőség ellenőrzését.

Konfigurációkezelő rendszerek: A konfigurációkezelő rendszerek, mint például az Ansible vagy a Puppet, lehetővé teszik a rendszer konfigurációjának automatizálását, a beállítások és konfigurációk nyomon követését és az infrastruktúra felügyeletét.

Ezek az eszközök lehetővé teszik a hatékonyabb implementációs nézőpont tervezését és megvalósítását, ami elengedhetetlen az alkalmazások hatékony működéséhez és fejlesztéséhez.



6.2 Szoftver Architektúra Típus

A szoftver architektúra típusok olyan általános architekturális mintázatok, amelyeket a szoftverfejlesztési projektekben alkalmaznak. Az architektúra típusok különböző szempontok alapján csoportosíthatók, például az alkalmazás architektúrája, a rendszerarchitektúra vagy a szolgáltatásorientált architektúra szempontjából.

Szoftver architektúra Típusok

Többrétegű (layered) architektúra: Az alkalmazás különböző rétegeit különválasztják, hogy azok elkülönüljenek egymástól, és az egész alkalmazást könnyebb legyen kezelni, karbantartani és tesztelni.

Komponens-alapú (component-based) architektúra: Az alkalmazás különböző komponensekre van bontva, amelyek függetlenek egymástól, és a fejlesztők az egyes komponenseket külön-külön kezelhetik.

Mikroszolgáltatások (microservices) architektúra: Az alkalmazás egy sor független szolgáltatásból áll, amelyek önállóan működnek és kommunikálnak egymással.

Szolgáltatás-orientált (service-oriented) architektúra: Az alkalmazás funkciói szolgáltatásokként vannak megvalósítva, amelyeket más alkalmazások is felhasználhatnak.

Adatközpontú (data-centric) architektúra: Az alkalmazás adatokra épül, és az adatok központi szerepet játszanak az alkalmazás tervezésében és fejlesztésében.

Esemény vezérelt (event-driven) architektúra: Az alkalmazás az eseményekre reagál, és az események vezérlik az alkalmazás működését és viselkedését.

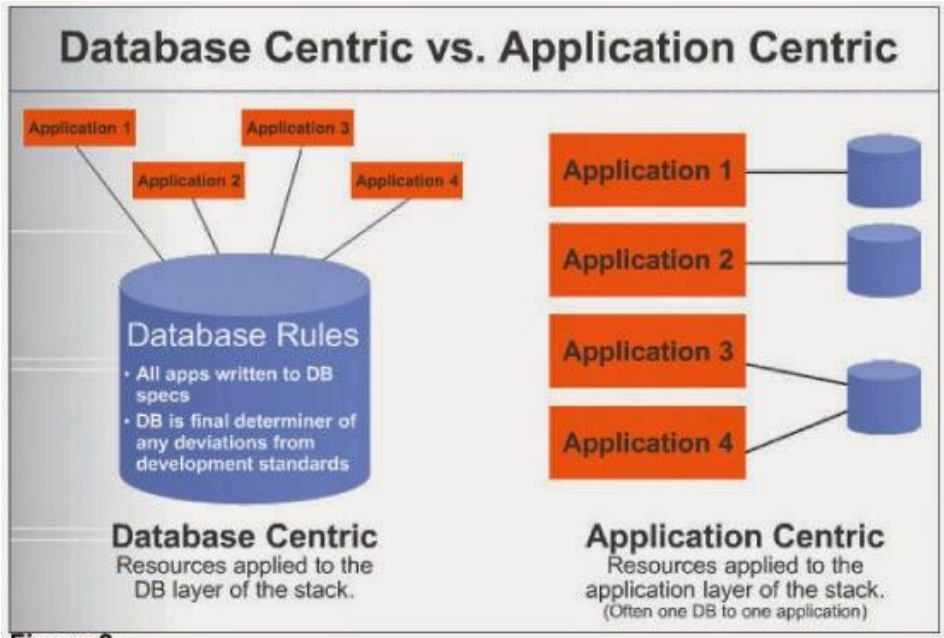


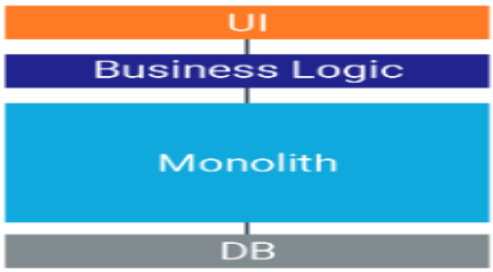
Figure 2

6.3 Szoftver Architektúra Stílus

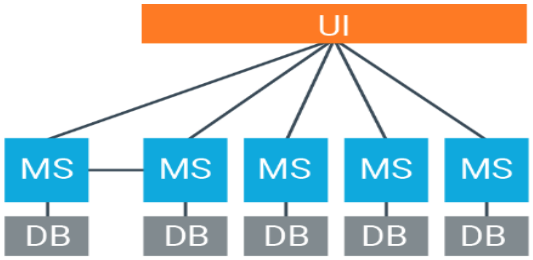
A szoftverarchitektúra-stílus olyan megközelítés, amely meghatározza, hogyan épül fel az alkalmazás vagy rendszer architektúrája. **Az architektúra-stílusok általában magukban foglalnak szabályokat, irányelveket, tervezési mintákat és protokollokat, amelyek segítenek az alkalmazások és rendszerek hatékony tervezésében, fejlesztésében, tesztelésében és karbantartásában**.

Szoftverarchitektúra Stílusok

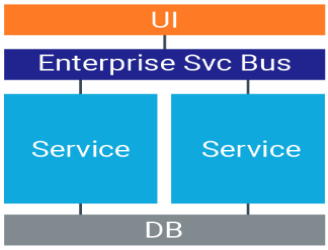
Monolitikus architektúra: A monolitikus architektúra a szoftverek hagyományos megközelítését alkalmazza. Ebben az architektúrában az alkalmazás egységesen van összekapcsolva, és az összes része egyetlen kódbázisban található. Ez a megközelítés általában a kisebb méretű alkalmazásokhoz és rendszerekhez alkalmazható.



Mikroszolgáltatások: A mikroszolgáltatások az alkalmazások és rendszerek felbontását javasolják kisebb, önállóan működő komponensekre, amelyek a szolgáltatásokon keresztül kommunikálnak egymással. Ez a megközelítés lehetővé teszi az alkalmazások és rendszerek könnyebb skálázhatóságát és karbantarthatóságát.



Szolgáltatás-orientált architektúra: A szolgáltatás-orientált architektúra (SOA) olyan architektúra-stílus, amely a szolgáltatások összességére helyezi a hangsúlyt az alkalmazásokban és rendszerekben. Az SOA architektúrák kis, önálló szolgáltatásokra épülnek, amelyek könnyen újrahasznosíthatóak és összekapcsolhatóak más szolgáltatásokkal.



Többrétegű architektúra: A többrétegű architektúra olyan architektúra-stílus, amely az alkalmazások és rendszerek felbontását javasolja különféle rétegekre, amelyek a különböző funkciókat vagy szolgáltatásokat végzik. Általában a többrétegű architektúrák különválasztják az adatokat, a felhasználói felületet, az üzleti logikát és az adatkezelést.

6.3.1 Szoftver Architektúra Stílus - Objektumorientált (OO)

Az objektumorientált szoftver architektúra stílus (Object Oriented Software Architecture Style) az objektumorientált programozás (OOP) szemléletére épül, amely a szoftvertervezésre és -fejlesztésre szolgáló paradigmák egyike. Az objektumorientált programozás lehetővé teszi az alkalmazások fejlesztését objektumokra és osztályokra épülő hierarchikus rendszerben, amelyeket a szoftvertervezés során terveznek és implementálnak.

Az objektumorientált szoftver architektúra stílusának alapvető elemei az objektumok és az osztályok, amelyeket az alkalmazás adott részén hoznak létre. Az objektumok reprezentálják a rendszer belső állapotát és viselkedését, míg az osztályok az objektumok általánosabb tulajdonságait és viselkedését írják le. Az objektumorientált szoftver architektúra stílusa lehetővé teszi az alkalmazás moduláris felépítését, a részek egymástól függetlenül, de egymással együttműködve működnek, és könnyen bővíthetők, karbantarthatók és tesztelhetők.

Az objektumorientált szoftver architektúra stílusának előnyei közé tartozik a szoftvertervezési folyamat hatékonysága, a rugalmasság, az újrafelhasználhatóság és a karbantarthatóság növelése. Az objektumorientált szoftver architektúra stílusa általában jól skálázható, könnyen érthető és egyszerűbb hibakeresést tesz lehetővé.

Az objektumorientált szoftver architektúra stílusa ma már széles körben használatos, és a legtöbb objektumorientált programozási nyelv és keretrendszer támogatja ezt a szemléletet.

6.4 Szoftver Architektúra Minta (Pattern)

A szoftver mintázatok olyan általános tervezési megoldások, amelyeket ismételten alkalmazhatunk a szoftvertervezés során. **A szoftver mintázatok segítségével azonos tervezési problémákra egy megszokott megoldást alkalmazhatunk, amely bizonyítottan hatékony és jól bevált.**

A szoftver mintázatok általában tervezési sablonok, amelyek leírják a problémát, amelyre megoldást kínálnak, és azt, hogyan lehet ezt a megoldást alkalmazni a szoftver tervezésében.

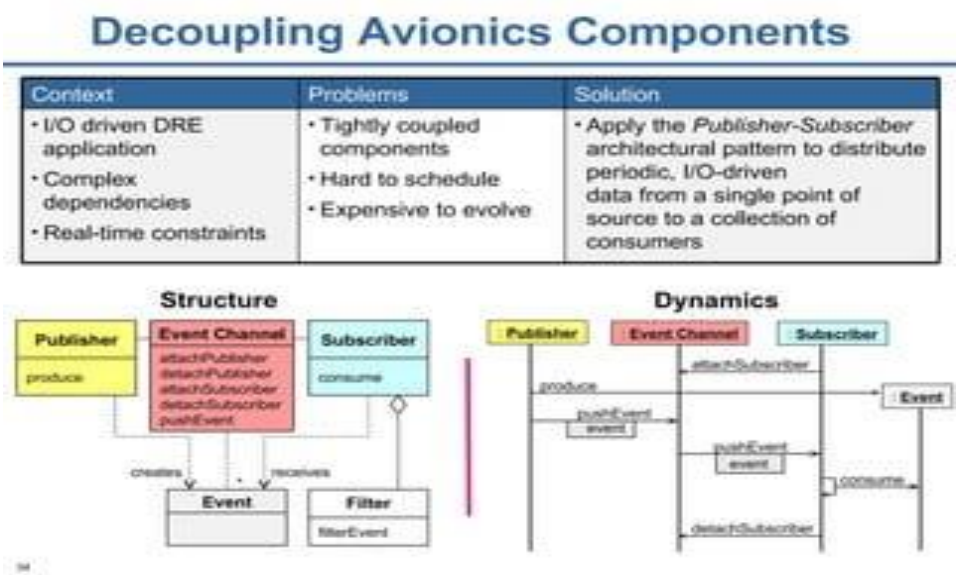
Szoftver Architektúra Minta sablon

Leírják a problémát: A szoftver mintázatok leírják a problémát, amelyet meg kell oldani, és az ezzel kapcsolatos megkötéseket.

Megadnak egy megoldási sablont: A szoftver mintázatok megadják az ismételten alkalmazható megoldási sablont, amely hatékonyan megoldja a problémát.

Adnak példákat: A szoftver mintázatok gyakran példákat adnak arra, hogyan lehet alkalmazni a mintázatot valós szoftvertervezési problémákra.

A szoftver mintázatok előnye, hogy bizonyítottan hatékonyak és jól beváltak a szoftvertervezésben. Az ismételt alkalmazásuk csökkenti a hibák és az időveszteség kockázatát, és segít az egységes, jól strukturált és karbantartható kódbázis létrehozásában. Néhány példa a szoftver mintázatokra a Singleton, az Observer, a Factory, az Adapter stb.



6.4.1 Tervezési Minta (Pattern)

A tervezési minták olyan ismétlődő tervezési megoldások, amelyeket azért hoztak létre, hogy javítsák az alkalmazás vagy rendszer tervezési minőségét, hatékonyságát és karbantarthatóságát. A minták olyan megoldásokat kínálnak, amelyeket már teszteltek, bizonyítottak és megbízhatóak a tervezési problémákra adott válaszként.

A minták lehetnek általánosak vagy specifikusak, és különböző területeken használhatók, mint például a szoftvertervezés, az adatmodellezés, az adattárház tervezés, az üzleti folyamat tervezés, stb. Az általánosabb minták olyan tervezési megoldásokat kínálnak, amelyeket széles körben lehet alkalmazni, míg a specifikus minták olyan tervezési megoldásokat kínálnak, amelyeket egy adott területen használnak.

A tervezési minták általában három részből állnak:

- **a probléma leírásából,**
- **a megoldás leírásából**
- **és a következmények leírásából.**

Az egyes minták leírják a problémát, amelyet megoldanak, és hogyan kell alkalmazni őket a megoldáshoz. Az egyes minták előnyei közé tartozik a hatékonyabb szoftvertervezés, a könnyebb karbantarthatóság, az olvashatóság és az újrafelhasználhatóság.

Tervezési minta példák

Model-View-Controller (MVC) pattern: az alkalmazások tervezésére alkalmazott általános pattern, amely megkülönbözteti a modellt (adatok), a nézetet (felhasználói felület) és az irányítót (vezérlő).

Singleton minta: csak egy példányt engedélyez az osztályból. Ez a minta hatékonyabbá teszi az objektumok kezelését és segít a memória használat optimalizálásában.

Facade pattern: egy egységes interfészt biztosít az összetett rendszerekhez, hogy könnyebben kezelhetőek legyenek.

Adapter pattern: lehetővé teszi, hogy az egyik interfész használható legyen egy másik interfésszel.

Factory minta: A Factory minta egy olyan tervezési minta, amely segít az objektumok létrehozásában és inicializálásában. Ez a minta különösen hasznos, amikor az objektumok dinamikusan hozzák létre és inicializálják.

Observer minta: Az Observer minta egy olyan tervezési minta, amely lehetővé teszi az objektumok és a rendszer közötti kommunikáció követését. Az Observer minta figyeli az objektumok változásait, és értesíti a rendszert az eseményekről.

Ezek az általános tervezési minták segítenek a szoftvertervezőknek és a szoftverfejlesztőknek a hatékonyabb és strukturáltabb szoftvertervezésben. A tervezési minták alkalmazása lehetővé teszi a hatékonyabb és jobb minőségű alkalmazások és rendszerek létrehozását.

6.4.2 Tervezési ellen-Minta (anti-Pattern)

A tervezési anti-Patternek olyan megoldások vagy tervezési minták, amelyek rosszul működnek, és gyakran negatív hatásokkal járnak az alkalmazások vagy rendszerek tervezésére és fejlesztésére. Az anti-Patternek gyakran azokból a tervezési megoldásokból vagy mintákból erednek, amelyeknek nem sikerült megfelelően megbirkózniuk az adott tervezési problémával.

Az anti-Patternek lehetnek tervezési hibák, túl bonyolult vagy nem optimális tervezési megoldások, vagy egyszerűen csak rosszul működő tervezési minták. Az antipatternek hátrányai közé tartozik a lassú és nehézkes fejlesztési folyamat, a nehéz karbantarthatóság és a hosszabb időtartamú fejlesztési időszakok.

Néhány tervezési antipattern

Megállapodás hiánya: Ha a csapat nem határozza meg egyértelműen a feladatokat és az elvárásokat, akkor előfordulhat, hogy a fejlesztési folyamat során a feladatok nem teljesülnek megfelelően.

Túl bonyolult megoldás: Ha a tervezők túl bonyolult tervezési megoldásokat választanak, akkor a fejlesztési folyamat lassabbá és nehézkesebbé válhat, valamint az alkalmazás vagy rendszer karbantarthatósága is csökkenhet.

„Sárkupac” ("Big Ball of Mud"): olyan alkalmazás vagy rendszer tervezése, amely rendkívül bonyolult és nem összefüggő kód bázison alapul, amelynek karbantartása és továbbfejlesztése nagyon nehéz.

Elavult technológia használata: Ha a tervezők régi vagy elavult technológiákat használnak, akkor az alkalmazás vagy rendszer nem lesz hatékony, és a továbbfejlesztése is nehézkessé válhat.

Az antipatternek felismerése és elkerülése kulcsfontosságú a hatékony és eredményes szoftvertervezés és fejlesztés érdekében. Az antipatternek elkerülése és helyes tervezési megoldások alkalmazása lehetővé teszi a hatékonyabb és jobb minőségű alkalmazások és rendszerek létrehozását.

6.5 Szoftver Architektúra Keretrendszer

A szoftver architektúra keretrendszer (Software Architecture Framework) egy olyan eszköz, amely előre meghatározott struktúrákat, irányelveket és modelleket tartalmaz a szoftver architektúra tervezéséhez és dokumentálásához. A keretrendszer általában előre elkészített sablonokat, szabályokat és eszközöket tartalmaz a szoftverarchitektúra tervezési folyamatának egyszerűsítésére.

A szoftver architektúra keretrendszerek általában segítséget nyújtanak az architektúra tervezése során a következő területeken:

- **Az alkalmazás moduláris architektúrájának megtervezése**
- **Az alkalmazás komponensek közötti kapcsolatok megtervezése és dokumentálása**
- **Az alkalmazás folyamatmodelljének megtervezése**
- **Az alkalmazás adatmodelljének megtervezése**
- **Az alkalmazás hardverarchitektúrájának megtervezése**
- **Az alkalmazás tesztstratégiájának megtervezése**

A szoftver architektúra keretrendszerek használatának előnyei közé tartozik a szoftverarchitektúra hatékonyabb tervezése és dokumentálása, az egységes megközelítés alkalmazása, valamint a szabványosított megoldások és eszközök használata. A szoftver architektúra keretrendszerek segítségével a szoftverarchitektúra tervezése egyszerűbbé és strukturáltabbá válik, ami hozzájárul a fejlesztési folyamat hatékonyabbá és eredményesebbé tételéhez.

Néhány példa szoftver architektúra keretrendszerekre:

- **TOGAF** (The Open Group Architecture Framework)
- **Zachman Framework**
- **RM-ODP** (Reference Model for Open Distributed Processing)
- **ISO/IEC 42010:2011** (Systems and software engineering -- Architecture description)

Ezek a keretrendszerek általában kiterjedt dokumentációkat, eszközöket, sablonokat és szabályokat tartalmaznak a szoftverarchitektúra tervezéséhez és dokumentálásához. A megfelelő szoftver architektúra keretrendszer kiválasztása az adott szoftverfejlesztési projekt céltól és követelményektől függ.

6.6 Architektúra Stílus vs. Architektúra Nézőpont

Az architektúra stílus és az architektúra nézőpont eltérő módon írják le az alkalmazások és rendszerek architektúráját.

Az **architektúra stílus** azokat az architekturális mintákat írja le, amelyek meghatározzák, hogy az alkalmazás vagy rendszer részei hogyan kommunikálnak egymással és milyen módon vannak egymással kapcsolatban. Az architektúra stílusok magukban foglalják azokat a tervezési döntéseket, amelyek hatással vannak az alkalmazás vagy rendszer teljesítményére, skálázhatóságára és karbantarthatóságára. Néhány példa az architektúra stílusokra az objektumorientált, szolgáltatás-orientált, adatelemző, esemény-vezérelt stb.

Az **architektúra nézőpont** azokat az architekturális szempontokat írja le, amelyek az alkalmazás vagy rendszer adott részeire fókuszálnak. Az architektúra nézőpontok meghatározzák, hogy milyen információkra van szükség a rendszer vagy alkalmazás adott részének tervezése során. Néhány példa az architektúra nézőpontokra a logikai nézőpont, a fizikai nézőpont, a felhasználói nézőpont, a folyamat nézőpont stb.

Tehát az architektúra stílus azokat az architekturális mintákat írja le, amelyek hatással vannak az alkalmazás vagy rendszer működésére és teljesítményére, míg az architektúra nézőpont azokat az architekturális szempontokat írja le, amelyek az alkalmazás vagy rendszer adott részére fókuszálnak. Mindkét megközelítés fontos szerepet játszik az alkalmazás vagy rendszer hatékony tervezésében és fejlesztésében.

7 Technológiai eszköztár

Olyan eszközök, platformok és alkalmazások, amelyek segítik az információkat gyűjteni, tárolni, feldolgozni és megosztani az üzleti folyamatok támogatása érdekében.

Az információs rendszerek technológiái átfogóak és változatosak lehetnek, és számos területet magukba foglalnak, mint például az adatbázis-kezelés, az adatfeldolgozás, a szoftvertervezés és fejlesztés, a hálózati kommunikáció, az adatvédelem és az adatbiztonság.

Információs rendszerek technológiái

- **Adatbázis-kezelő rendszerek**: Oracle, MySQL, Microsoft SQL Server, PostgreSQL stb.
- **Big Data technológiák**: Hadoop, Spark, Hive, Pig, MapReduce stb.
- **Felhőszolgáltatások**: AWS, Microsoft Azure, Google Cloud stb.
- **Alkalmazásfejlesztési platformok**: Java, .NET, Python, Ruby, PHP stb.
- **Hálózati kommunikációs protokollok**: TCP/IP, HTTP, FTP, DNS, SMTP, SNMP stb.

Az információs rendszerek technológiái az üzleti folyamatok hatékonyabbá és hatékonyabbá tételében játszanak fontos szerepet, amelyek lehetővé teszik az adatok és az információk hatékonyabb felhasználását a vállalkozások számára. Az információs rendszerek technológiái folyamatosan fejlődnek és változnak, így az üzleti igényeknek és a technológiai trendeknek való megfelelés érdekében folyamatosan figyelni kell az új fejleményekre és lehetőségekre.

7.1 Adatkezelési technológiák

Az adatkezelési technológiák olyan eszközök és módszerek, amelyek segítségével az adatokat hatékonyan lehet kezelni, tárolni, feldolgozni, elemzik és megosztani. Ezek a technológiák az adatok gyűjtésétől kezdve azok tárolásán és feldolgozásán át egészen az adatok elemzéséig és értelmezéséig terjednek.

A legfontosabb adatkezelési technológiák

1. **Adatbázis-kezelő rendszerek** (DBMS): Az adatbázis-kezelő rendszerek olyan szoftverek, amelyek segítségével az adatokat hatékonyan lehet tárolni, kezelni és keresni adatbázisokban.

2. **Adatintegrációs technológiák**: Az adatintegrációs technológiák olyan eszközök és módszerek, amelyek segítségével az adatokat különböző forrásokból integrálhatják és összekapcsolhatják, például az adatbázisokból, fájlokból, webes szolgáltatásokból vagy IoT eszközökből.

3. **Adatfeldolgozási technológiák**: Az adatfeldolgozási technológiák olyan eszközök és módszerek, amelyek segítségével az adatokat automatikusan lehet feldolgozni és elemezni, például a gépi tanulás, a mesterséges intelligencia, az adatelemzés és a nagy adathalmazok feldolgozása.

4. **Adatbiztonsági technológiák**: Az adatbiztonsági technológiák olyan eszközök és módszerek, amelyek segítségével az adatokat biztonságosan lehet tárolni, védeni és megosztani, például a tűzfalak, az adatvédelmi szabályzatok, a titkosítás és az azonosítási és hitelesítési rendszerek.

5. **Adatkatalógusok**: Az adatkatalógusok olyan eszközök, amelyek segítségével az adatokat és azok metaadatait dokumentálni, nyilvántartani és rendszerezni lehet. Az adatkatalógusok elősegítik az adatok újrafelhasználhatóságát és csökkentik az adatok redundanciáját.

6. **Adatvizualizációs eszközök**: olyan eszközök, amelyek segítségével az adatokat grafikus formában jeleníthetjük meg, és ezzel könnyebben érthetővé tehetjük azokat.

7. **Adatelemzési technológiák**: olyan eszközök és módszerek, amelyek segítségével az adatokat elemezhetjük, értékelhetjük és megérthetjük azokat, például adatbányászati és gépi tanulási módszerekkel.

7.1.1 Adatfeldolgozás technológiák

Az adatfeldolgozási technológiák olyan eszközök, amelyek lehetővé teszik az adatok elemzését, átalakítását, tárolását és kezelését. A Big Data adatfeldolgozási módszer, technológia kifejezetten a nagy mennyiségű, valós idejű adatfeldolgozást támogatja, ami korunk legnagyobb kihívása.

Apache Hadoop: Az Apache Software Foundation által fejlesztett, nyílt forráskódú adatfeldolgozási keretrendszer, amely lehetővé teszi az adatok nagy adathalmazokban történő tárolását és elemzését.

Apache Spark: Az Apache Software Foundation által fejlesztett, nyílt forráskódú adatfeldolgozási keretrendszer, amely lehetővé teszi az adatok nagy sebességgel történő feldolgozását és elemzését.

Apache Kafka: Az Apache Software Foundation által fejlesztett, nyílt forráskódú, elosztott adatfolyam-kezelő rendszer, amely lehetővé teszi a valós idejű adatok feldolgozását és elemzését.

Apache Storm: Az Apache Software Foundation által fejlesztett, nyílt forráskódú adatfeldolgozási keretrendszer, amely lehetővé teszi a valós idejű adatok feldolgozását és elemzését.

Apache NiFi: Az Apache Software Foundation által fejlesztett, nyílt forráskódú adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását és átalakítását.

Talend: Az adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását különböző forrásokból, beleértve az adatbázisokat, fájlokat és webes szolgáltatásokat.

Amazon Web Services (AWS): A felhőalapú szolgáltatásokat kínáló platform, amely lehetővé teszi az adatok tárolását, feldolgozását és elemzését az AWS kínálatában szereplő különböző szolgáltatások segítségével.

Flink: Az Apache Software Foundation által fejlesztett nyílt forráskódú adatfeldolgozási platform, amely lehetővé teszi a valós idejű adatfeldolgozást és elemzést.

Cassandra: Az Apache Software Foundation által fejlesztett elosztott adatbázis-kezelő rendszer, amely lehetővé teszi az adatok gyors feldolgozását és elemzését.

Elasticsearch: Az Elasticsearch BV által fejlesztett nyílt forráskódú keresőmotor és analitikai platform, amely lehetővé teszi a nagy mennyiségű adat hatékony keresését és elemzését.

Snowflake: A Snowflake Computing által fejlesztett felhőalapú adatfeldolgozási platform, amely lehetővé teszi a nagy mennyiségű adat hatékony feldolgozását és elemzését.

Ezek az adatfeldolgozási technológiák széles körben használatosak a vállalati környezetben és az adatintenzív alkalmazásokban az adatok hatékony feldolgozására és elemzésére. Az adatfeldolgozási technológiák lehetővé teszik az adatok kezelését és elemzését a valós idejű döntéshozatalhoz és az üzleti folyamatok optimalizálásához.

7.1.2 Adatbázis-kezelő rendszerek

Az adatbázis rendszerek technológiái olyan eszközök, szoftverek, platformok és protokollok, amelyek az adatok hatékony és biztonságos tárolását, kezelését, lekérdezését és feldolgozását teszik lehetővé az adatbázisokban. Ide tartoznak az adatbázis-kezelő rendszerek, a tranzakciós feldolgozó rendszerek, a nagy adatrendszerek, a NoSQL adatbázisok, a kolonnos adatbázisok, az objektum-relációs leképzések, az adattárházak és az adatbányászati rendszerek. Az adatbázis rendszerek technológiái számos területen hasznosak lehetnek, mint például az üzleti intelligencia, a döntéstámogatás, az adatintegráció és az adatvizualizáció.

Az adatbázis-kezelő rendszerek olyan szoftverek, amelyek lehetővé teszik az adatok hatékony és biztonságos tárolását, kezelését és lekérdezését.

Oracle Database: Az Oracle Corporation által fejlesztett relációs adatbázis-kezelő rendszer, amely lehetővé teszi az adatok hatékony tárolását és kezelését.

Microsoft SQL Server: A Microsoft által fejlesztett relációs adatbázis-kezelő rendszer, amely lehetővé teszi az adatok hatékony tárolását és kezelését.

MySQL: Az Oracle Corporation által fejlesztett, nyílt forráskódú relációs adatbázis-kezelő rendszer, amely lehetővé teszi az adatok hatékony tárolását és kezelését.

PostgreSQL: A PostgreSQL Global Development Group által fejlesztett, nyílt forráskódú objektum-relációs adatbázis-kezelő rendszer, amely lehetővé teszi az adatok hatékony tárolását és kezelését.

MongoDB: A MongoDB Inc. által fejlesztett dokumentum-orientált adatbázis-kezelő rendszer, amely lehetővé teszi az adatok hatékony tárolását és kezelését.

Cassandra: Az Apache Software Foundation által fejlesztett, elosztott adatbázis-kezelő rendszer, amely lehetővé teszi az adatok hatékony tárolását és kezelését.

Redis: Az Redis Labs által fejlesztett, memória-alapú adatbázis-kezelő rendszer, amely lehetővé teszi az adatok gyors tárolását és kezelését.

Ezek az adatbázis-kezelő rendszerek különböző típusú adatok hatékony tárolását és kezelését teszik lehetővé, és széles körben használják a vállalati környezetben és az adatintenzív alkalmazásokban.

7.1.3 Adattárház támogató rendszerek

Az adattárház technológiát támogató adatbázis rendszerek számos lehetőséget kínálnak az adattárházak és az adatelemzési folyamatok támogatására.

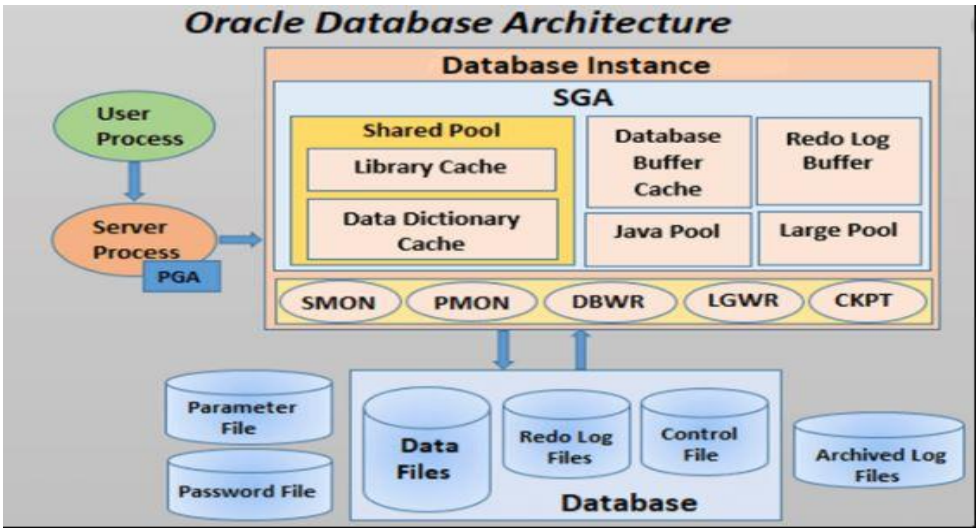
Microsoft SQL Server: Az SQL Server adatbázis rendszer számos lehetőséget kínál az adattárházak létrehozására és az adatelemzési folyamatok támogatására, beleértve az integrált adatelemzési szolgáltatásokat, az adattárházépítőt és a Power BI adatelemzési szoftvert.

Oracle Database: Az Oracle Database rendszer az adattárházak és az adatelemzési folyamatok támogatására különféle lehetőségeket kínál, például az Oracle Warehouse Builder-t, az Oracle Business Intelligence-t és az Oracle Data Mining-et.

IBM DB2: Az IBM DB2 adatbázis rendszer adattárházak és adatelemzési folyamatok támogatására tervezett funkciókat kínál, beleértve a DB2 Warehouse Manager-t és az IBM Cognos Business Intelligence szoftvert.

SAP HANA: Az SAP HANA egy olyan memóriában tárolt adatbázis rendszer, amely az adattárházak és az adatelemzési folyamatok támogatására készült. Az SAP HANA magában foglalja a SAP Business Warehouse-t és az SAP Lumira adatelemzési szoftvert.

Ezek az adatbázis rendszerek különféle lehetőségeket kínálnak az adattárházak és az adatelemzési folyamatok támogatására, és lehetővé teszik a hatékonyabb adatfeldolgozást és elemzést. Az adattárház technológiáját támogató adatbázis rendszerek alkalmazása lehetővé teszi a jobb minőségű és hatékonyabb adatelemzést és döntéshozatalt a vállalkozások számára.



7.1.4 Adatkatalógusok

Az adatkatalógusok az adatokkal kapcsolatos információk gyűjteményei, amelyek lehetővé teszik az adatok megtalálását, értelmezését és felhasználását.

Adatkatalógusok

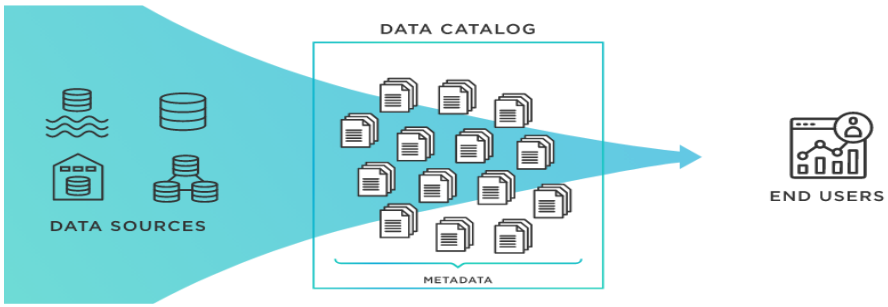
Amazon Web Services Glue Data Catalog: Az Amazon Web Services adatkatalógus megoldása, amely lehetővé teszi a felhő alapú adatforrások és adatfeldolgozási szolgáltatások kezelését.

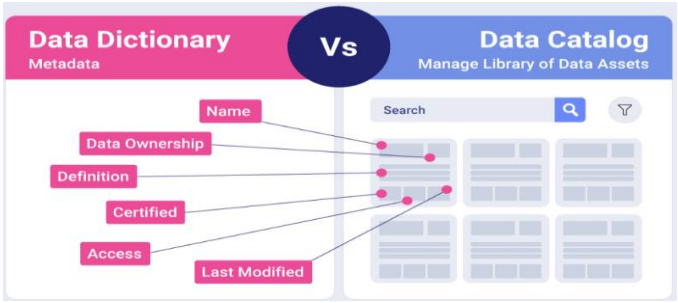
IBM Watson Knowledge Catalog: Az IBM Watson adatkatalógusa, amely lehetővé teszi a strukturált és struktúrátlan adatok gyűjtését, címkézését és kategorizálását.

Apache Atlas: Az Apache Software Foundation által fejlesztett nyílt forráskódú adatkatalógus megoldás, amely lehetővé teszi az adatok metaadatokkal való címkézését és kategorizálását.

Collibra: Az adatok kezelésére szakosodott vállalat adatkatalógus megoldása, amely lehetővé teszi a vállalati adatok kezelését és az adatok metaadatokkal való címkézését.

Google Cloud Data Catalog: A Google Cloud adatkatalógusa, amely lehetővé teszi a felhő alapú adatforrások és adatfeldolgozási szolgáltatások kezelését.





7.2 Integrációs technológiák

Az integrációs technológiák olyan eszközök és megoldások, amelyek segítségével különböző informatikai rendszereket és alkalmazásokat lehet összekötni és adatokat cserélni közöttük.

Integrációs módszerek

1. **Adatbázis replikáció**: A replikáció során az adatokat másolják az egyik adatbázisból a másikba, és a két adatbázis szinkronizált marad. Ez a módszer hatékony, ha az adatok áramlása egyszerű és folyamatos.
2. **Web Services modell**: A Web Services modell egy szabványosított protokollt és interfészt biztosít a rendszerek közötti kommunikációhoz. A SOAP (Simple Object Access Protocol) és a REST (Representational State Transfer) két példa ilyen interfészre.
3. **Message-oriented Middleware (MOM) modell**: Az MOM modell egy üzenetalapú architektúrát biztosít a rendszerek közötti kommunikációhoz. Az üzenetek közvetlenül a rendszerek közötti kapcsolat nélkül érkeznek a middleware-hez, amely az üzeneteket továbbítja a megfelelő rendszereknek.
4. **Middleware**: A middleware olyan szoftverréteg, amely lehetővé teszi az alkalmazások közötti kommunikációt, adatcserét és integrációt. A middleware segítségével az alkalmazások könnyedén csatlakoztathatók más rendszerekhez és alkalmazásokhoz.
5. **Adatintegrációs platformok**: Az adatintegrációs platformok olyan szoftverek, amelyek lehetővé teszik az adatok összegyűjtését és összekapcsolását különböző forrásokból, és lehetővé teszik az adatok egységes kezelését és elemzését.
6. **ETL (extract, transform, load) eszközök**: Az ETL eszközök olyan szoftverek, amelyek lehetővé teszik az adatok kinyerését (extract) különböző forrásokból, az adatok átalakítását (transform) egy egységes formába, és az adatok betöltését (load) egy célrendszerbe vagy adatbázisba.
7. **Hub and Spoke modell**: Az integráció központi hub köré épül, amely összeköti a különböző alkalmazásokat és szolgáltatásokat. A hub (központi csomópont) lehet fizikai vagy virtuális, és lehetővé teszi az adatok átvitelét a hubon keresztül az összes csatlakoztatott alkalmazás és szolgáltatás között.
8. **Enterprise Service Bus (ESB) modell**: Az ESB egy szoftveres komponens, amely összekapcsolja az alkalmazásokat és szolgáltatásokat. Az ESB lehetővé teszi az adatok átvitelét az összes csatlakoztatott alkalmazás és szolgáltatás között.

9. **Point-to-Point modell**: Az adatok közvetlenül továbbítódnak két alkalmazás között. Ez a megoldás egyszerű és könnyen implementálható, azonban a skálázhatóság problémáit veti fel, és nehezebben kezelhető, ha az integrációs pontok száma nő.
10. **Representational State Transfer (REST) modell**: A REST modell egy könnyű súlyú integrációs megközelítés, amely az HTTP protokollra épül. A REST API-kat használják az adatok közötti kommunikációra, és lehetővé teszik az alkalmazások számára az adatok olvasását, írását és frissítését.
11. **Cloud integráció** : folyamat, amely során különböző rendszerek, alkalmazások vagy adatforrások integrálódnak egy vagy több felhőalapú szolgáltatásba. A felhőintegráció célja a különböző adatforrások és alkalmazások egységesítése és összekapcsolása, valamint az adatok szabad áramlása és megosztása.

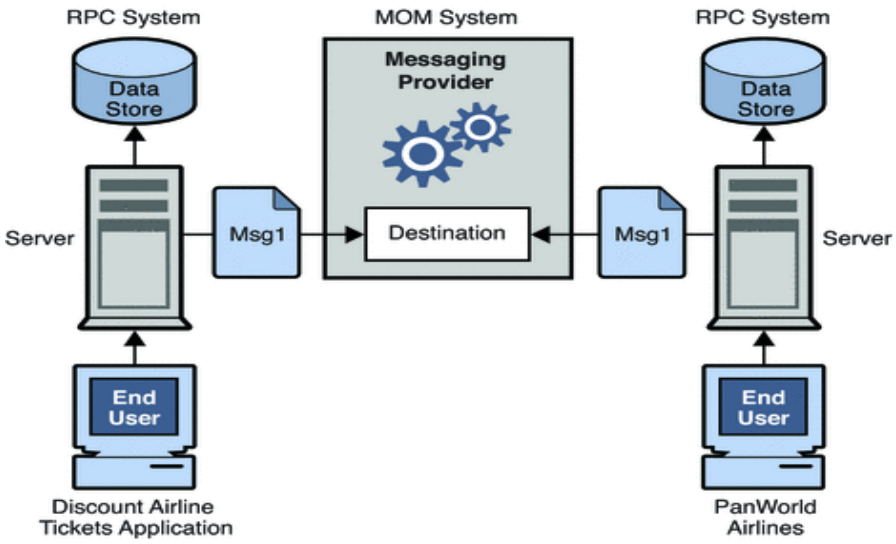
7.2.1 Általános integráció – Üzenetközvetítő réteg modell

Message Oriented Middleware (MOM) egy olyan szoftveres közvetítő rendszer, amely a számítógépes hálózaton keresztül biztosítja a kommunikációt és az adatátvitelt. Az üzenetek az alkalmazások közötti kommunikáció alapegységei, amelyek különböző formátumban lehetnek, például szöveges, bináris vagy XML formátumban.

Az MOM központi eleme a központi üzenetküldő (Message Broker), amely felelős az üzenetek fogadásáért, tárolásáért és továbbításáért a célalkalmazások felé. **Az alkalmazások nem közvetlenül kommunikálnak egymással, hanem a központi üzenetküldőn keresztül.**

Az üzenetek lehetnek szinkron vagy aszinkron típusúak. A szinkron üzenetek küldése és fogadása azonnali választ igényel a fogadó alkalmazástól. Az aszinkron üzenetek küldése és fogadása nem igényel azonnali választ, az üzeneteket a központi üzenetküldő tárolja, majd továbbítja a fogadó alkalmazásnak.

Az MOM előnyei közé tartozik az alkalmazások közötti kommunikáció standardizálása, a független alkalmazások közötti kommunikáció biztosítása, a nagy adatforgalom kezelése, a megbízható és skálázható adatátvitel és az üzenetek biztonságos kezelése. Azonban az MOM rendszerek nagy méretű és összetettségűek lehetnek, és magas üzemeltetési költséggel járhatnak.



7.2.2 Adatintegráció platformok

Az adatintegrációs platformok olyan szoftveres eszközök és megoldások, amelyek lehetővé teszik az adatok összekapcsolását és integrálását különböző forrásokból. Az adatintegrációs platformok használata lehetővé teszi az adatok automatikus integrálását, ami gyorsabb és hatékonyabb adatfeldolgozást eredményez.

Funkcionalitás

Adatforrások összekapcsolása: Az adatintegrációs platformok lehetővé teszik az adatforrások összekapcsolását, amely lehetővé teszi az adatok integrálását különböző adatforrásokból.

Adatátalakítás: Az adatintegrációs platformok segítségével az adatokat egységes formátumra lehet átalakítani, ami lehetővé teszi az adatok egyszerűbb integrálását.

Adatfeldolgozás: Az adatintegrációs platformok lehetővé teszik az adatok feldolgozását, amely lehetővé teszi az adatok szűrését, csoportosítását és aggregálását.

Adattárolás: Az adatintegrációs platformok lehetővé teszik az adatok tárolását különböző adatbázisokban vagy adattárházakban.

Automatizált integráció: Az adatintegrációs platformok lehetővé teszik az adatok automatikus integrálását, amely segíti a folyamatos és hatékony adatfeldolgozást.

Adatellenőrzés és hibakezelés: Az adatintegrációs platformok lehetővé teszik az adatok ellenőrzését és hibakezelését, amely biztosítja az adatok pontosságát és megbízhatóságát.

Adatintegrációs platformok

Az adatintegrációs platformok számos vállalat és szervezet számára elérhetők.

Microsoft Power BI: Az adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását, átalakítását és elemzését.

Talend: Az adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását különböző forrásokból, beleértve az adatbázisokat, fájlokat és webes szolgáltatásokat.

IBM InfoSphere Information Server: Az adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását különböző forrásokból, beleértve az adatbázisokat, fájlokat, webes szolgáltatásokat és adattárházakat.

Oracle Data Integrator: Az adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását és integrálását különböző forrásokból.

SnapLogic: Az adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását különböző forrásokból, beleértve az adatbázisokat, fájlokat, webes szolgáltatásokat és felhőszolgáltatásokat.

MuleSoft Anypoint Platform: Az adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását különböző forrásokból, beleértve az adatbázisokat, fájlokat, webes szolgáltatásokat és felhőszolgáltatásokat.

Dell Boomi: Az adatintegrációs platform, amely lehetővé teszi az adatok összekapcsolását különböző forrásokból, beleértve az adatbázisokat, fájlokat, webes szolgáltatásokat és felhőszolgáltatásokat.

7.2.3 Adatintegráció technológiák

Az adatintegrációs technológiák olyan eszközök és módszerek, amelyek lehetővé teszik az adatok összekapcsolását és integrálását különböző forrásokból.

Az adatintegrációs technológiák hasznosak az adatok egyszerűbb és hatékonyabb kezelésében, és segítenek az adatok értékének növelésében és az üzleti döntések megalapozásában.

Adatintegrációs technológiák

Adattárházak: Az adattárházak olyan nagy adatbázisok, amelyekben az adatokat különböző forrásokból integrálják, átalakítják és rendszerezik. Az adattárházak segítségével az adatokat egyszerűen lehet lekérdezni és elemzeni.

ETL (Extract, Transform, Load) szoftverek: Az ETL szoftverek segítségével az adatokat különböző forrásokból kinyerik (Extract), átalakítják (Transform) és betöltik (Load) az adatbázisokba vagy adattárházakba.

Adatintegrációs platformok: Az adatintegrációs platformok olyan eszközök, amelyek lehetővé teszik az adatok integrálását különböző forrásokból, például adatbázisokból, fájlokból, webes szolgáltatásokból vagy IoT eszközökből. Az adatintegrációs platformok segítségével az adatokat automatizálva lehet integrálni és összekapcsolni.

Adatvirtuálizáció: Az adat virtualizáció olyan technológia, amely lehetővé teszi az adatok virtuális integrálását, ami azt jelenti, hogy az adatokat különböző forrásokból virtuálisan integrálják és jelenítik meg az alkalmazások számára, miközben az eredeti források nem módosulnak.

Ezek az adatintegrációs technológiák segítenek az adatok integrálásában és az adatok felhasználásának hatékonyságában, miközben az adatok megbízhatóságát és konzisztenciáját is biztosítják. Az adatintegrációs technológiák segítenek az üzleti folyamatok optimalizálásában, az adatok pontosságának növelésében és az üzleti döntések megalapozásában.

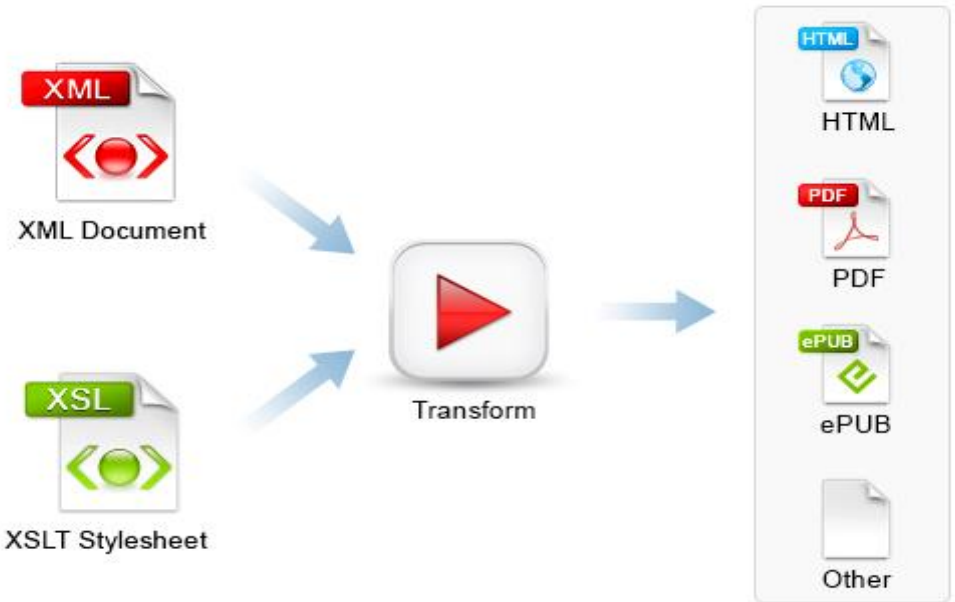
7.2.4 Adatintegráció – XML modell

Az XML (Extensible Markup Language) technológia az integráció szempontjából rendkívül fontos. Az XML egy univerzális adatcserére használt formátum, amely lehetővé teszi a különböző rendszerek közötti adatcserét. Az XML-t általában strukturált adatok leírására használják, amelyek különböző rendszerek közötti kommunikációra szolgálnak, például az adatok importálására vagy exportálására más rendszerekbe.

Az XML fontos jelentőséggel bír az integrációban, mert lehetővé teszi a különböző rendszerek közötti adatcserét. Az XML formátum lehetővé teszi, hogy a különböző alkalmazások és rendszerek adatokat küldjenek és fogadjanak, függetlenül attól, hogy milyen programozási nyelvet vagy adatbázis-kezelő rendszert használnak. Az XML formátum általában rugalmasabb és sokoldalúbb, mint más adatcserére használt formátumok, és az XML-ben leírt adatokat könnyebb feldolgozni és átalakítani más formátumokba.

Az XML technológia további előnye az, hogy lehetővé teszi az adatok címzését, validálását és egyéb funkciókat, amelyek javítják az adatok pontos és hatékony átvitelét. Az XML-t használják webes szolgáltatások, az elektronikus kereskedelem és az adatintegrációs projektek terén is.

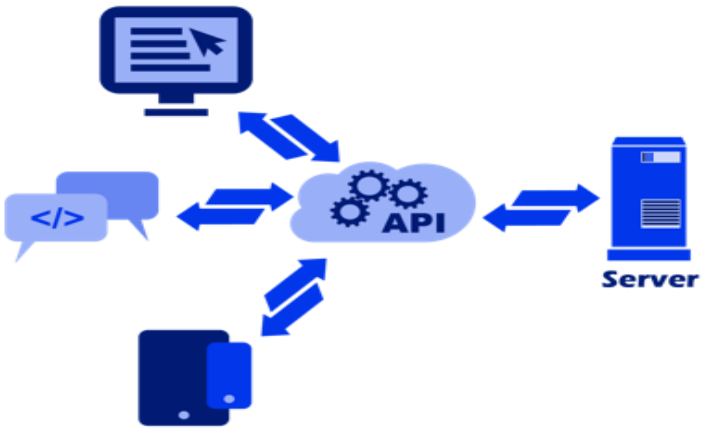
Összefoglalva, az XML fontos szerepet játszik az integrációban, mert lehetővé teszi a különböző rendszerek közötti adatcserét, és általános formátumként használható a különböző alkalmazások és rendszerek közötti adatok átvitelére.

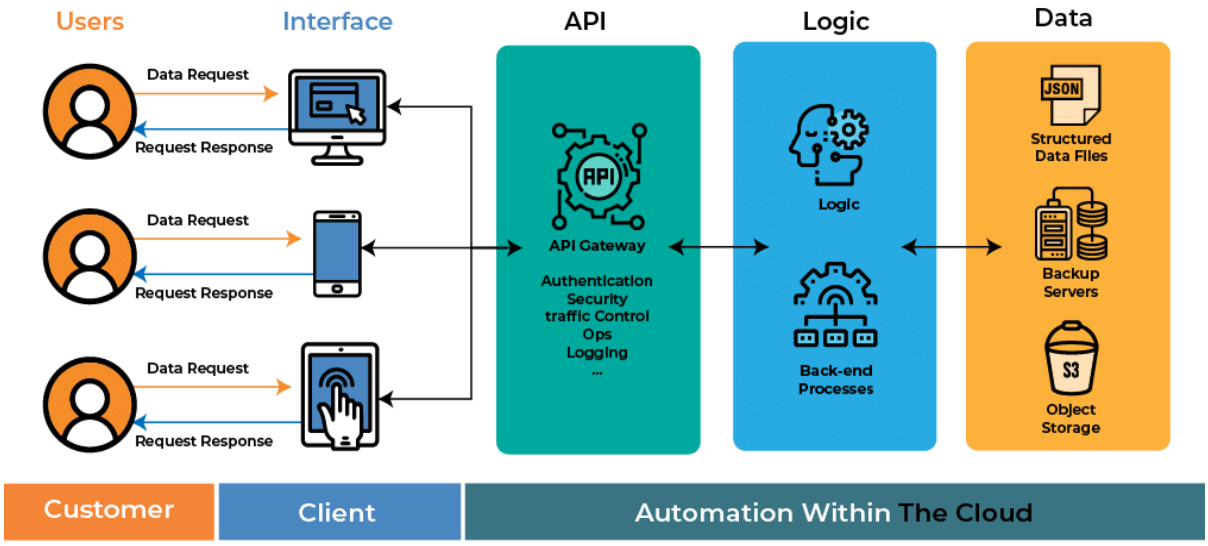


7.2.5 Adatintegráció - API modell

Az API-k hatékony eszközök arra, hogy a felhasználó számára megtalálják az adatokat, még akkor is, ha azok egy másik adatbázisban, alkalmazásban vagy programban találhatók. Számos népszerű alkalmazás lényegében portál a mélyebb információs adatbázisokba. A tartalom streaming-szolgáltatások (CDS), az időjárási alkalmazások és sok más alkalmazás API-k segítségével segítik a felhasználókat az összetett adatok elérésében.

Az API-k rövidítésekként szolgálhatnak, megvédve a felhasználót vagy a fejlesztőt a komplexitástól azáltal, hogy előre elkészített folyamatokat vagy kódot hívnak. Egy fejlesztőnek például nem kell értenie a műholdak vagy a helymeghatározási algoritmusok működéséhez ahhoz, hogy GPS-követéssel rendelkező alkalmazást készíthessen. Az API-k segítségével összekapcsolhatják alkalmazásukat egy összetettebb GPS-rendszerrel.





7.2.6 Adattár integráció – Adattárház (Data Warehouse)

Az adatintegráció és az adattárházak (**Data Warehouse**) kapcsolódik egymáshoz, hiszen az adatintegráció célja az adatok összehangolása és összehangolása az egész vállalat számára, míg az adattárház a létrehozott adatok fizikai tárolására szolgál, amelyet a felhasználók később jelentésekre és elemzésekre használhatnak fel.

Adattárház technológiák

1. **Adatmodellezés**: Az adattárházak adatmodelljei általában a csillag- és hópehely modell alapján készülnek, amelyek a dimenziók és tények hierarchikus összekapcsolására összpontosítanak.

2. **Adattisztítás és adattranszformáció**: Az adattárházba való integrálás előtt az adatokat megtisztítják és átalakítják, hogy azok az adattárház adatmodelljével kompatibilisek legyenek.

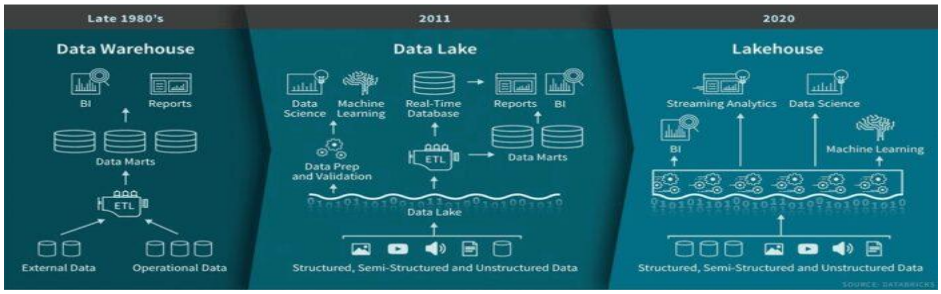
3. **Adattárolás**: Az adattárházak az adatokat rendszerint oszlopon alapuló adatbázisokban tárolják, amelyek optimalizáltak az aggregálásra és az elemzésekre.

4. **Adatintegráció**: Az adatokat az adattárházba különböző forrásokból, például tranzakciós rendszerekből, üzleti alkalmazásokból és más adatbázisokból integrálják.

5. **Adatfeldolgozás**: Az adattárházak gyakran tartalmaznak előre meghatározott adatfeldolgozási logikákat, például aggregálás, szűrés és számítások.

6. **Adatfelhasználás**: Az adattárházak által tárolt adatokat jelentésekhez, elemzésekhez és üzleti döntésekhez használják fel.

7. **Adatbiztonság**: Az adattárházak által tárolt adatok érzékenyek lehetnek, ezért fontos, hogy az adatbiztonsági előírásokat szigorúan betartsák az adattárházakban.



7.2.7 Szolgáltatás integráció – Szolgáltatás-orientált modell

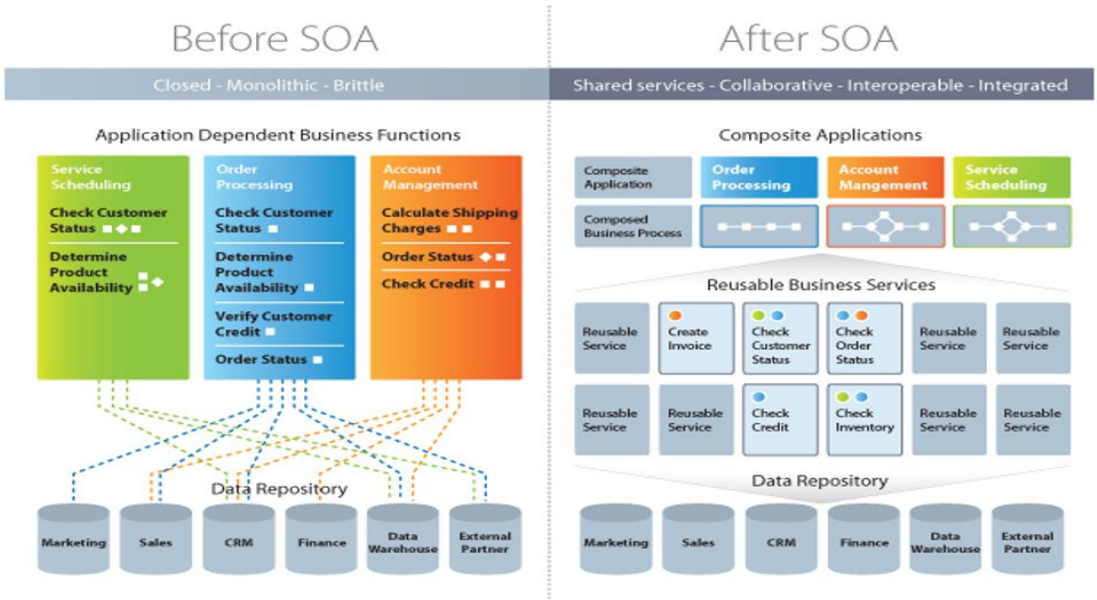
SOA (Service-Oriented Architecture) egy szoftvertervezési módszer, amely alapvetően a szolgáltatásokra összpontosít. A SOA-ra épülő alkalmazások a különböző szolgáltatásokat kombinálják és használják fel, hogy a felhasználók számára értékes, funkcionális rendszereket biztosítsanak.

A SOA előnyei közé tartozik, hogy csökkenti az alkalmazások közötti függőséget, javítja az alkalmazások rugalmasságát és újrafelhasználhatóságát, lehetővé teszi a különböző technológiák és platformok közötti integrációt, és könnyen bővíthetővé teszi a rendszert.

A SOA modellben az alkalmazásokat szolgáltatásokra bontják, amelyeket a rendszer belső vagy külső felhasználók használnak. A szolgáltatások különböző technológiákkal és platformokkal rendelkezhetnek, de egységes interfészekkel kommunikálnak egymással.

A SOA architektúrája három fő rétegből áll: az adatszolgáltatási réteg, a logikai szolgáltatási réteg és a felhasználói szolgáltatási réteg. Az adatszolgáltatási réteg biztosítja az adatok tárolását és hozzáférését, a logikai szolgáltatási réteg tartalmazza az üzleti logikát, a felhasználói szolgáltatási réteg pedig a felhasználói felületet és a felhasználói interakciókat.

A SOA-t használó vállalatok gyakran használnak szolgáltatásorientált infrastruktúrát (SOI), amely egy olyan eszközrendszer, amely lehetővé teszi az alkalmazások közötti kommunikációt és az adatok megosztását a szolgáltatások révén.

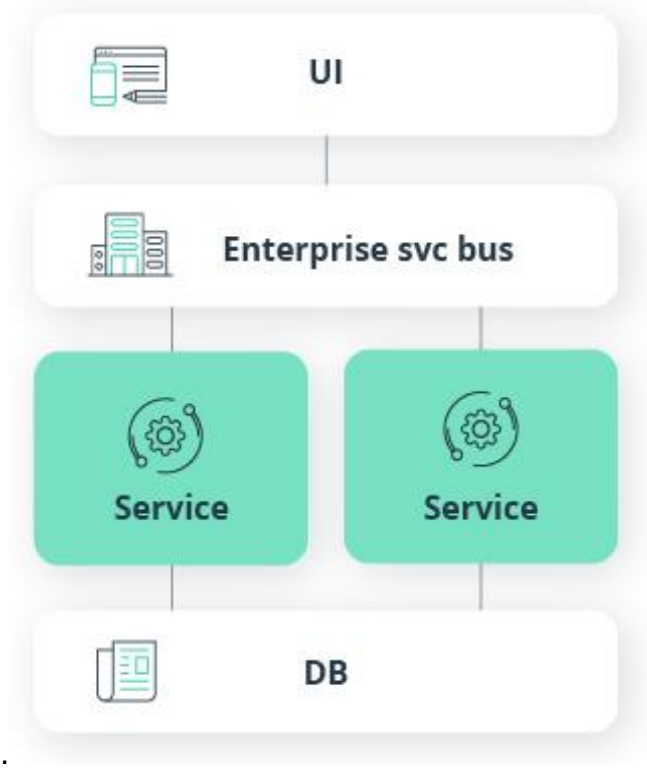


7.2.8 Alkalmazás integráció – Szolgáltatás busz modell

Az Enterprise Service Bus (ESB) egy integrációs modell, amely egy központi helyet biztosít az alkalmazások közötti kommunikációhoz. Az ESB egy köztes réteget hoz létre, amely lehetővé teszi az alkalmazások számára, hogy egymással kommunikáljanak, anélkül, hogy közvetlen kapcsolatba kellene lépniük egymással.

Az ESB modell az üzenetek közvetítésére és átalakítására szolgál az alkalmazások között. Az ESB az üzeneteket egy központi üzenetközponton keresztül továbbítja, ahol azokat az üzenetformátumokat átalakítják és validálják, amelyeket a fogadó alkalmazások használnak. Az ESB segítségével az üzeneteket a küldő alkalmazástól a fogadó alkalmazásig továbbíthatjuk anélkül, hogy az alkalmazások közvetlen kapcsolatba kellene lépniük egymással.

Az ESB modell előnye, hogy csökkenti az alkalmazások közötti integráció költségeit, javítja az alkalmazások közötti kommunikációt, és lehetővé teszi a szervezet számára, hogy rugalmasabban alkalmazkodjon az üzleti igényekhez. Az ESB modell hátránya, hogy komplex rendszert hoz létre, amely nehezen karbantartható és kezelhető, ha nincs megfelelő szaktudás az alkalmazások integrációjában



.

7.2.9 Internetes integráció – Webszolgáltatás modell

A webszolgáltatás egy szoftvertervezési módszer, amely lehetővé teszi a különböző rendszerek közötti kommunikációt az interneten keresztül. Egy webszolgáltatás egy olyan alkalmazás, amely lehetővé teszi a kliensek számára, hogy adatokat kérjenek és kapjanak választ egy szerverről. A webszolgáltatások lehetnek egyszerűek vagy összetettek, és lehetnek elérhetők helyi vagy távoli szervereken.

A webszolgáltatások általában a következőképpen működnek: egy kliens alkalmazás elküld egy kérést a szervernek, amely magában foglalja a kért adatokat és a kliens azonosítóját. A szerver megkapja a kérést, és választ küld vissza a kliensnek. A válasz általában tartalmazza az adatokat, amelyeket a kliens kért, valamint egy státuszkódot, amely jelzi, hogy a kérés sikeres volt-e vagy sem.

A webszolgáltatások legfontosabb előnyei a következők:

- **Interoperabilitás**: A webszolgáltatások lehetővé teszik a különböző rendszerek közötti kommunikációt, függetlenül attól, hogy milyen programozási nyelvet vagy operációs rendszert használnak.
- **Hálózati elérhetőség**: A webszolgáltatásokhoz való hozzáférés az interneten keresztül történik, így könnyen elérhetők bárhol a világon.
- **Újrafelhasználhatóság**: A webszolgáltatások újra felhasználhatok, így az alkalmazások különféle részei között is felhasználhatók.
- **Rugalmasság**: A webszolgáltatások különféle felhasználási esetekhez alkalmazkodnak, így könnyen illeszthetők a különböző alkalmazásokba.

A webszolgáltatások különböző protokollokat használhatnak, például SOAP (Simple Object Access Protocol), REST (Representational State Transfer) vagy XML-RPC (Remote Procedure Call). Az adatok átvitele általában XML vagy JSON (JavaScript Object Notation) formátumban történik.

7.3 Alkalmazási rendszerek

Az alkalmazási rendszerek olyan szoftverek vagy programok összessége, amelyek egy adott feladatot vagy üzleti folyamatot hajtanak végre. Az alkalmazási rendszerek lehetnek kliens-szerver alapúak, amelyekben a számítógépek közötti feladatmegosztás történik, vagy lehetnek webalkalmazások, amelyek az interneten keresztül érhetők el. Az alkalmazási rendszerek lehetnek irodai szoftverek, például szövegszerkesztők vagy táblázatkezelők, de lehetnek ipari alkalmazások is, például vezérlő rendszerek vagy adatgyűjtő rendszerek. Az alkalmazási rendszerek tervezése és fejlesztése során fontos szempontok a megbízhatóság, biztonság, teljesítmény és felhasználói élmény. Az alkalmazási rendszerek fejlesztése magában foglalja a tervezést, a programozást, a tesztelést és az implementálást.

1. **Ügyviteli rendszerek**: Ez az olyan rendszerek, amelyek az üzleti folyamatokat támogatják, például az értékesítés, a pénzügy és a szállítás területén. Ilyen rendszerek például a CRM (ügyfélkapcsolati menedzsment), a számlázó és a logisztikai rendszerek.
2. **Termelésirányítási rendszerek**: A gyártási folyamatok és a termelési erőforrások (munkaerő, gépek, alapanyagok) koordinálását támogató rendszerek. Ilyen rendszerek például a MES (gyártási végrehajtási rendszer) és az MRP (anyagkövető rendszer).
3. **Adatbázis-kezelő rendszerek**: Adataink tárolására és kezelésére szolgáló rendszerek, amelyek lehetnek relációs adatbázisok, NoSQL adatbázisok és adattárolási felhőszolgáltatások.
4. **Adatelemzési rendszerek**: Az adatok elemzésére és jelentéskészítésre szolgáló rendszerek, amelyek lehetnek BI (Business Intelligence) rendszerek, adatbányászati rendszerek és adatvizualizációs eszközök.
5. **Banki alkalmazási rendszer**: Ez az alkalmazási rendszer a banki folyamatok automatizálására szolgál, beleértve a pénzügyi tranzakciókat, az ügyfélszolgálatot és az értesítéseket. Az alkalmazási rendszer magában foglalhat webes felhasználói felületeket, adatbázisokat, integrációs interfészeket és más funkciókat.
6. **Egészségügyi alkalmazási rendszer**: Az egészségügyi alkalmazási rendszerek az egészségügyi folyamatokat támogatják, mint például a betegnyilvántartás, az orvosi diagnózis és az elektronikus egészségügyi nyilvántartások (EHR) kezelése. Az alkalmazási rendszerek tartalmazhatnak adatbázisokat, integrációs interfészeket és más komponenseket.
7. **Vállalatirányítási rendszer**: Az ERP (Enterprise Resource Planning) rendszerek a vállalatirányítást automatizálják és integrálják az összes vállalati folyamatot,

beleértve a pénzügyi számvitelt, a humánerőforrás-kezelést és a beszerzési folyamatokat. Az ERP rendszereknek általában adatbázisuk, felhasználói felületeik és integrációs interfészeik vannak.

8. **Logisztikai alkalmazási rendszer**: A logisztikai alkalmazási rendszerek a szállítási folyamatokat és az árukészlet-kezelést támogatják, beleértve a raktárkezelést, a szállítási menetrendeket és az áru nyomon követését. Az alkalmazási rendszerek tartalmazhatnak adatbázisokat, integrációs interfészeket és más funkciókat.
9. **E-kereskedelmi alkalmazási rendszer**: Az e-kereskedelmi alkalmazási rendszerek az online értékesítést és az ügyfélszolgálatot támogatják, beleértve a webes felhasználói felületeket, a fizetési folyamatokat és az értékesítési adatok nyomon követését. Az alkalmazási rendszereknek általában adatbázisuk, integrációs interfészeik és más funkcióik vannak.
10. **Elektronikus közszolgáltatási rendszerek**: Az ilyen jellegű rendszerek elsődleges célja a jogszabályi előírások szerinti közszolgáltatások nyújtása. Az ilyen rendszerek tervezésénél az a jellemző, hogy annak ellenére, hogy az ügyfelek ügyintézést könnyítik meg, nem elsődlegesen ügyfél igények felmérése alapján kell a megtervezni őket, hanem figyelembe kell venni a jogszabályi előírásokat is.

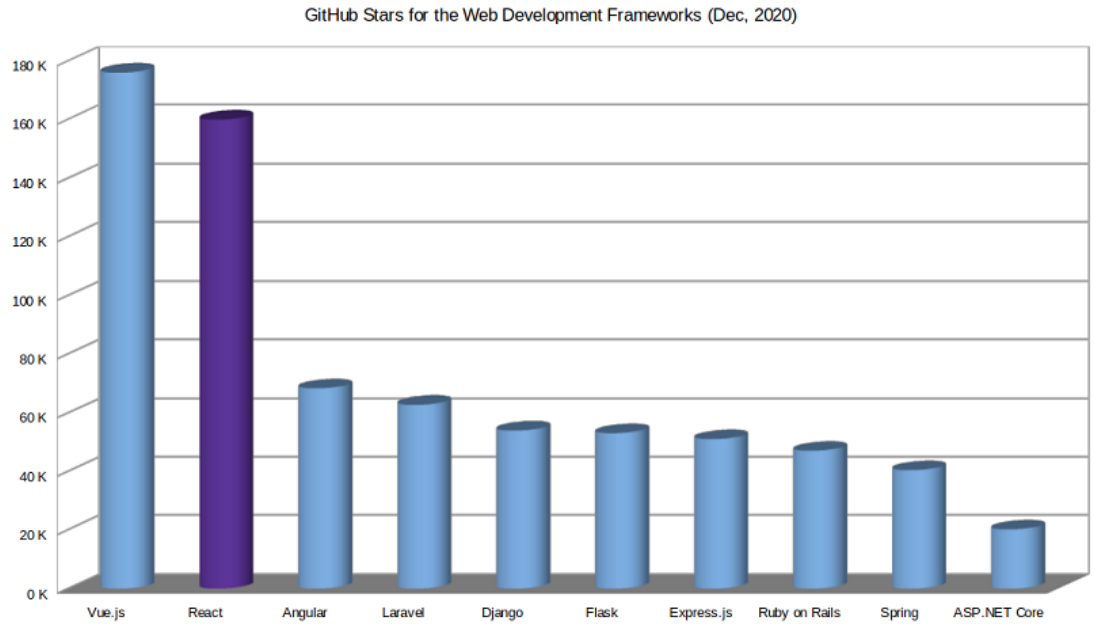
7.3.1 Alkalmazási rendszer technológiák

Az alkalmazási rendszer technológiák azok a technológiák, amelyeket alkalmaznak az alkalmazások tervezéséhez, fejlesztéséhez, üzemeltetéséhez és karbantartásához. Ezek a technológiák az alkalmazások különböző rétegeiben működnek, például a felhasználói felületen, az adatbázis-kezelésben, az üzleti logikában és a hálózati kommunikációban.

Alkalmazási rendszer technológia típusok

- **Felhasználói felület technológiák**: HTML, CSS, JavaScript, Angular, React, Vue.js
- **Adatbázis-kezelés technológiák**: SQL Server, MySQL, Oracle, PostgreSQL, Cassandra
- **Üzleti logika technológiák**: Java, .NET, Python, Ruby, PHP
- **Kommunikációs technológiák**: TCP/IP, HTTP, REST, SOAP, WebSockets

Az alkalmazási rendszer technológiák kiválasztása és használata nagyban függ az alkalmazás igényeitől és az üzleti követelményektől. Az alkalmazási rendszer technológiák gyorsan változnak és fejlődnek, így fontos az aktuális trendek és legjobb gyakorlatok figyelemmel kísérése és alkalmazása is.



7.3.1.1 Felhasználó felület – Angular

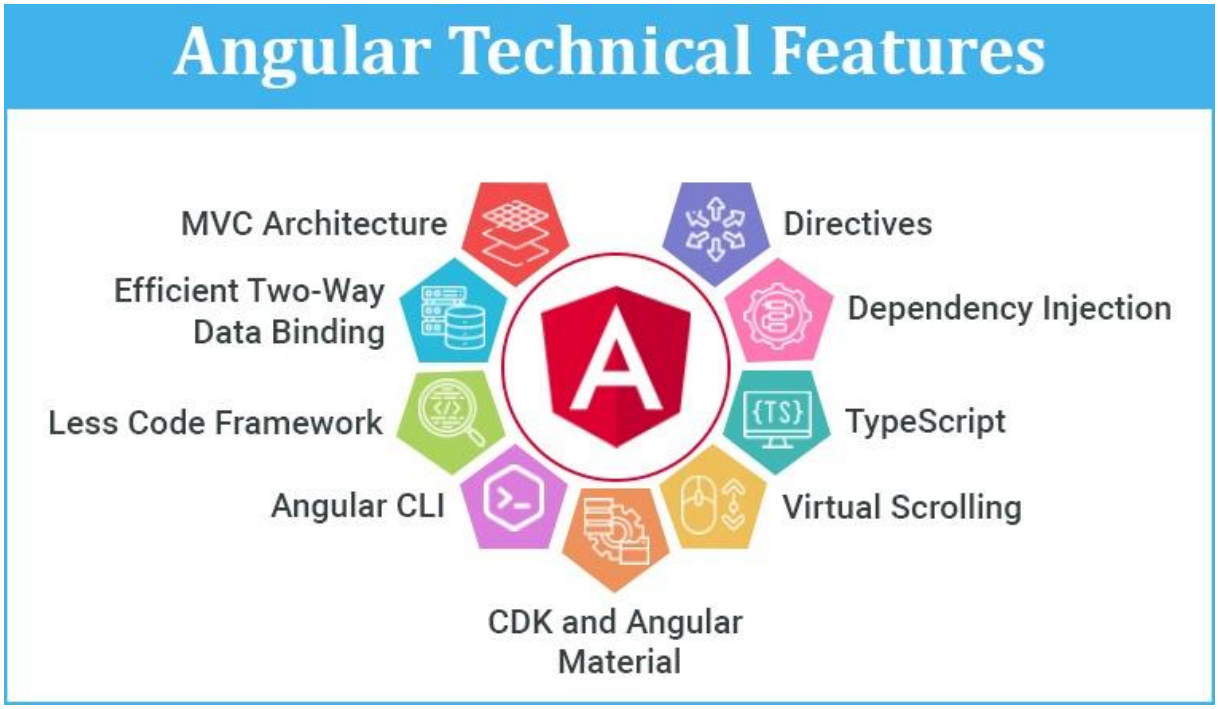
Angular egy ingyenes és nyílt forrású webes keretrendszer, amelyet a Google fejlesztett ki és karbantart.

A keretrendszer segítségével dinamikus, egyoldalas alkalmazások (Single Page Applications) készíthetők.

Az Angular alkalmazások TypeScript nyelven íródnak, és nagy hangsúlyt fektetnek az alkalmazások felhasználói felületének kialakítására és a kliens-oldali adatkezelésre.

Az Angular architektúrája komponens alapú, amely lehetővé teszi a kód újrafelhasználhatóságát és karbantarthatóságát.

Az Angular az egyik legnépszerűbb keretrendszer a modern webfejlesztésben, és számos nagy vállalat, például a Google, a Microsoft és az IBM is használja az alkalmazásaik fejlesztéséhez.



7.3.1.2 Felhasználó felület – React JS

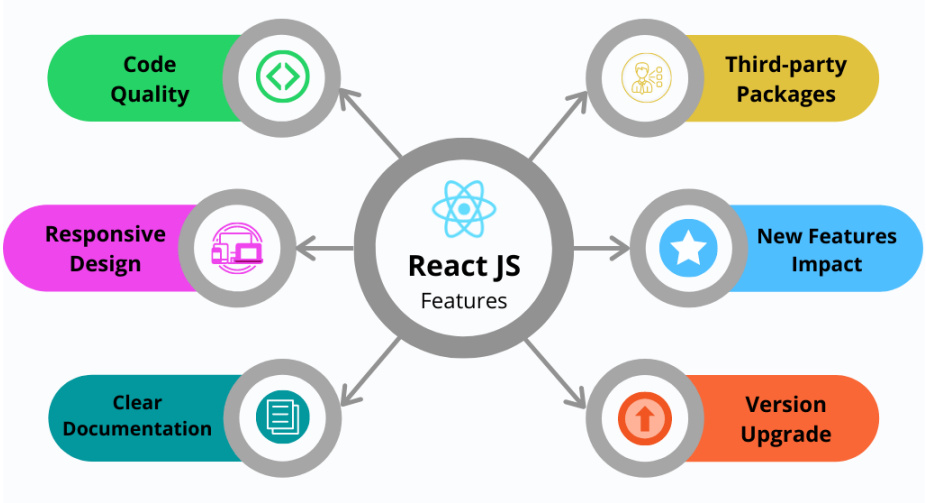
React egy nyílt forráskódú JavaScript könyvtár, amelyet felhasználói felületek (UI) készítésére használnak.

A React az egyik legnépszerűbb UI könyvtárak közé tartozik, és a Facebook fejlesztette ki. A React segítségével újra felhasználható és könnyen karbantartható felhasználói felületeket lehet készíteni.

A React alapja a komponenseknek, amelyek különböző funkciókat és tulajdonságokat tartalmaznak, és egymással kommunikálnak, hogy létrehozzanak egy komplett felhasználói felületet.

A React támogatja a JSX-t (JavaScript XML), amely lehetővé teszi a HTML-t hasonló kódszintaxis használatát a JavaScript-ben.

A React alkalmazásokat általában egyéb könyvtárakkal és keretrendszerekkel kombinálják, például a Redux-szal vagy a Next.js-sel.



7.3.1.3 Felhasználó felület – Swing

Swing egy Java-ban íródott grafikus felhasználói felületi (GUI) keretrendszer.

A Swing fejlesztése a 90-es évek végén kezdődött és a Java Foundation Classes (JFC) részeként jelent meg.

A Swing lehetővé teszi a fejlesztők számára a keresztplatformos GUI alkalmazások fejlesztését, amelyek ugyanúgy néznek ki és viselkednek, függetlenül attól, hogy melyik operációs rendszeren futnak.

A Swing tartalmazza a grafikus elemeket (gombok, szövegmezők, címke stb.) és azokat az eseménykezelőket, amelyek segítségével az alkalmazás reagál az felhasználói eseményekre (például a gombra kattintásra).

A Swing könnyen tanulható, és a Java nyelvhez hasonlóan objektumorientált paradigma alapján épül.

7.3.1.4 Felhasználó felület - Vue.js

Vue.js egy nyílt forráskódú JavaScript keretrendszer, amely lehetővé teszi a felhasználói felületek építését és a webes alkalmazások fejlesztését.

A Vue.js a komponens alapú architektúrán alapul, amely lehetővé teszi a fejlesztők számára, hogy elkülönítsék az alkalmazások különböző funkcióit és moduljait, és ezáltal egyszerűbbé tegyék azok kezelését és karbantartását.

A Vue.js könnyen tanulható és alkalmazható, mivel szintaxisa hasonló a HTML-hez, így egyszerűen beépíthető a már meglévő projektekbe is. Támogatja a deklaratív adatkötést, a dinamikus komponenseket, a felhasználói események kezelését és az animációkat is.

A Vue.js rugalmas és testre szabható, így lehetővé teszi a fejlesztők számára, hogy saját igényeik szerint formálják az alkalmazásokat.

7.3.1.5 Felhasználó felület - ASP.NET

ASP.NET egy keretrendszer a dinamikus webes alkalmazások és szolgáltatások fejlesztéséhez, amelyet a Microsoft fejlesztett ki. Az ASP.NET alapvetően az ASP (Active Server Pages) technológia utódja, és egy olyan modellre épül, amely lehetővé teszi a fejlesztők számára, hogy összekapcsolják a webes felhasználói felületet a szerveroldali adatokkal és üzleti logikával.

Az ASP.NET tartalmazza az ASP.NET Web Forms-t, amely lehetővé teszi a fejlesztők számára, hogy könnyedén létrehozzanak olyan webes alkalmazásokat, amelyek hasonlítanak a hagyományos asztali alkalmazásokra. Az ASP.NET MVC (Model-View-Controller) lehetővé teszi a fejlesztők számára, hogy szétválasszák a felhasználói felületet és az üzleti logikát, ami javítja az alkalmazások karbantarthatóságát és tesztelhetőségét.

Az ASP.NET Core egy új, áttervezett verziója az ASP.NET-nek, amely lehetővé teszi a fejlesztők számára, hogy cross-platform alkalmazásokat fejlesszenek Windows, Linux és macOS rendszereken. Az ASP.NET Core a .NET Core keretrendszeren alapul, amely gyorsabb és kevesebb erőforrást igényel, mint a korábbi .NET Framework.

Az ASP.NET a kiterjedt dokumentáció és a fejlesztői közösség támogatásával rendelkezik, és számos fejlesztői eszközt és bővítményt kínál, amelyek lehetővé teszik a fejlesztők számára, hogy gyorsan és hatékonyan fejlesszenek webes alkalmazásokat.

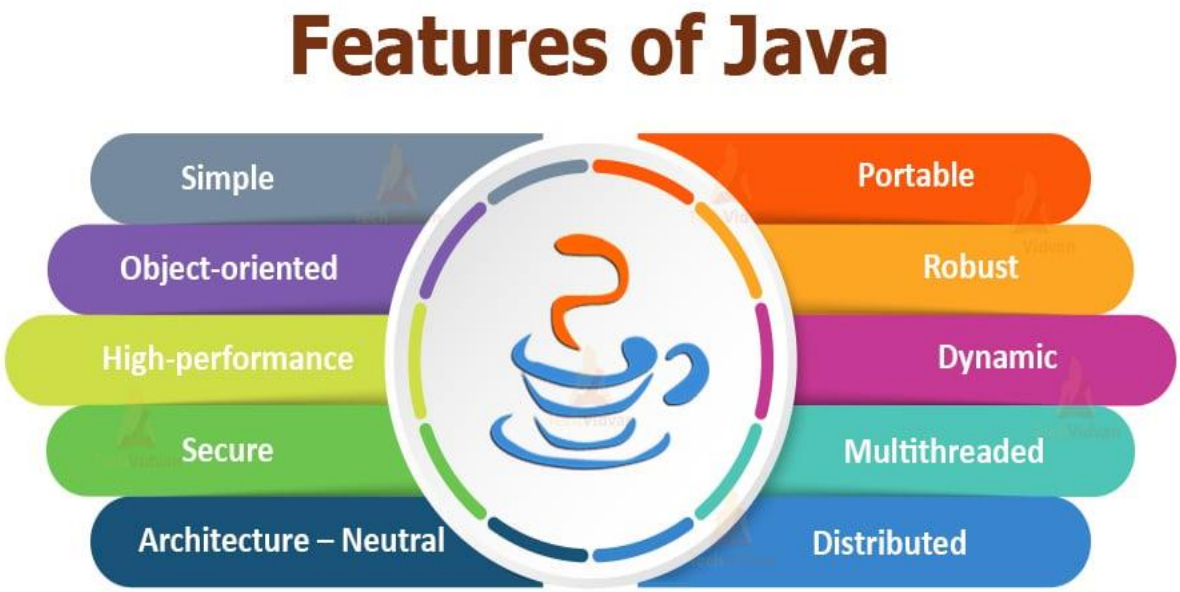
7.3.1.6 Üzleti logika – Java

Java egy objektumorientált, platformfüggetlen programozási nyelv és egy számítási platform, amely lehetővé teszi a programozók számára, hogy alkalmazásokat fejlesszenek a számítógépekhez, mobiltelefonokhoz, beágyazott eszközökhöz és az internethez is.

A Java nyelv jellemzője, hogy erősen típusos, nagyon széles körű használatra alkalmas, nagy kódbázisokhoz és fejlesztői közösségekhez kapcsolódik.

A Java platform számos API-t biztosít a fejlesztők számára, mint például a Java SE (Standard Edition), a Java EE (Enterprise Edition), a Java ME (Micro Edition) és a JavaFX.

A Java használata elterjedt a vállalati szoftverfejlesztésben, az Android mobilalkalmazások fejlesztésében, a szerveroldali webalkalmazások fejlesztésében és sok más területen.



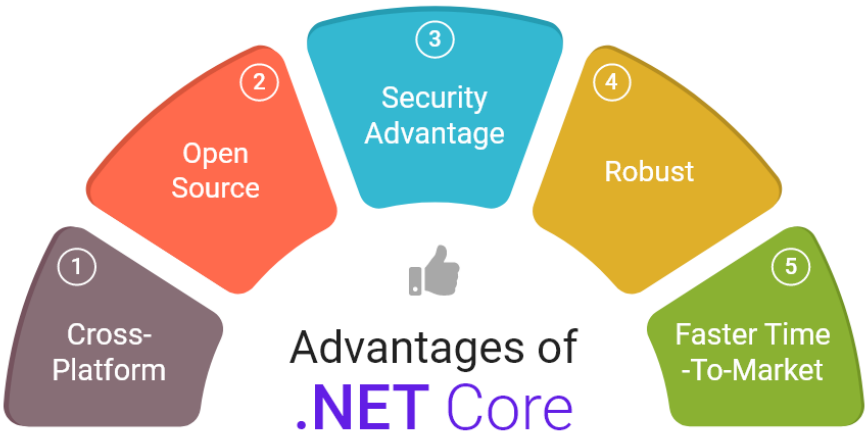
7.3.1.7 Üzleti logika – .NET

.NET (dotNET) a Microsoft által kifejlesztett keretrendszer, amely lehetővé teszi a Windows rendszeren futó alkalmazások, webszolgáltatások és mobilalkalmazások fejlesztését.

A .NET keretrendszer több programozási nyelvet támogat, mint például a C#, a Visual Basic .NET, a F# és az IronPython, és lehetővé teszi a programozók számára, hogy nagy hatékonysággal fejlesszenek korszerű alkalmazásokat a Windows platformon.

A .NET keretrendszerben számos komponens található, mint például az ASP.NET, a Windows Forms, az ADO.NET, a WPF és a WCF, amelyek segítségével a programozók különböző típusú alkalmazásokat fejleszthetnek.

A .NET keretrendszer nyílt forráskódú változata, a .NET Core, elérhető Windows, Linux és macOS operációs rendszereken is.



7.3.1.8 Üzleti logika – Python

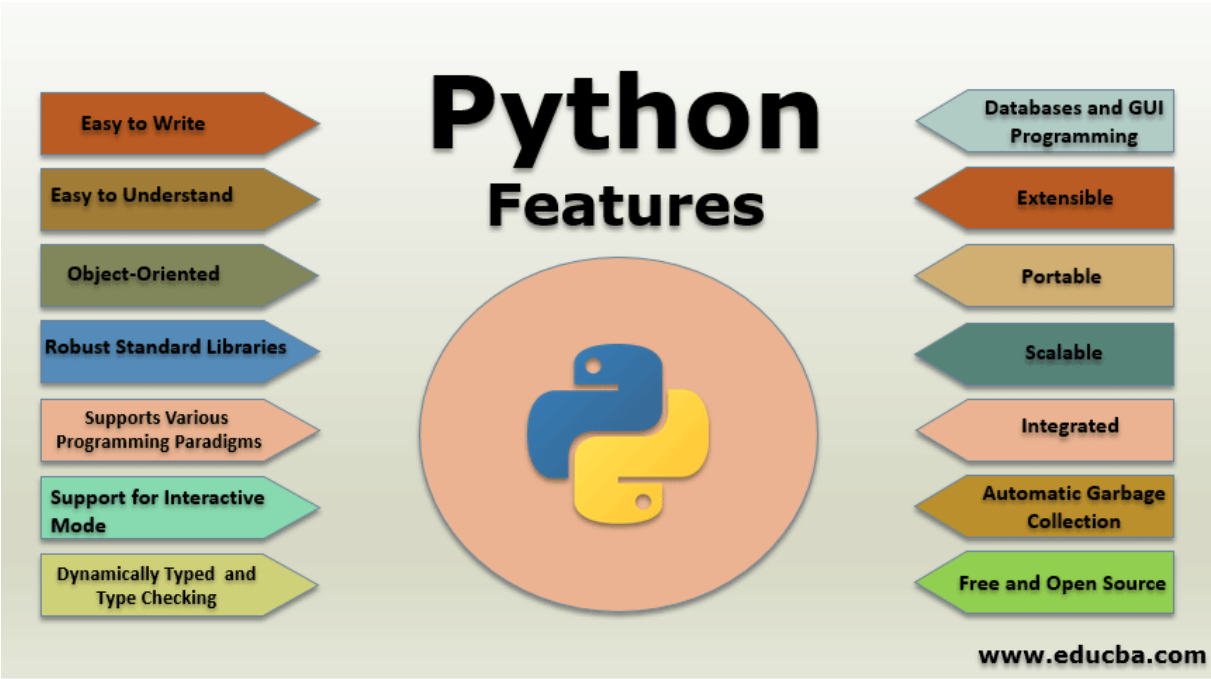
A Python egy magas szintű, értelmezett programozási nyelv, amely a kód olvashatóságára és egyszerű használatára helyezi a hangsúlyt.

Gyakran használják általános célú programozásra, tudományos számításokhoz, adatelemzéshez, mesterséges intelligenciához és gépi tanuláshoz, webfejlesztéshez és automatizálási feladatokhoz.

A Python legfontosabb jellemzői közé tartozik a dinamikus tipizálási rendszer, az automatikus memóriakezelés és a kiterjedt szabványkönyvtár.

Egyszerűségéről, sokoldalúságáról és közösségi támogatásáról is ismert, számos nyílt forráskódú könyvtár és keretrendszer áll rendelkezésre a különböző feladatokhoz.

A Python az utóbbi években az egyik legnépszerűbb programozási nyelvvé vált, amelyet fejlesztők és adattudósok egyaránt használnak.



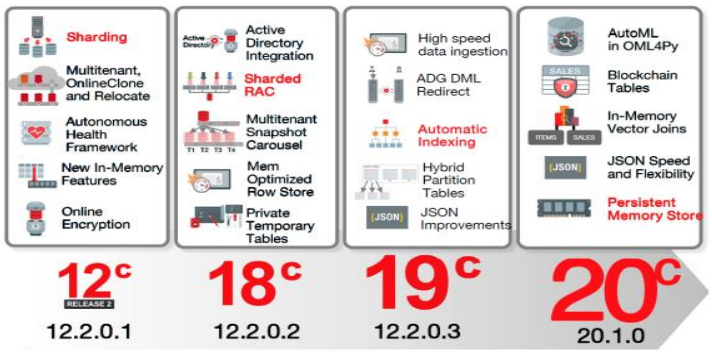
7.3.1.9 Adatbázis-kezelés – Oracle

Oracle egy adatbázis-kezelő rendszer és egyben cég is, amely az adatbázis-kezelés, az alkalmazásfejlesztés, a szervertechnológia és a felhőszolgáltatások területén tevékenykedik. Nagy teljesítményű, skálázható és megbízható adatbázis-kezelő rendszer, amely lehetővé teszi az adatok hatékony és biztonságos tárolását, kezelését és lekérdezését. Számos funkciót és szolgáltatást kínál, többek között tranzakciókezelést, adatintegritást, biztonságot és adatkapcsolatokat is.

Az Oracle rendszer az üzleti környezetben széles körben elterjedt, és számos nagyvállalat és szervezet használja az adatkezeléshez és az üzleti folyamatok támogatásához.

Oracle adatbázis jellemzők

- **Skálázhatóság**: Az Oracle adatbázis skálázható, így képes kezelni nagy mennyiségű adatot és megbízhatóan működni nagyobb üzleti alkalmazásokban.
- **Biztonság**: Az Oracle adatbázis tartalmaz biztonsági funkciókat, például felhasználói jogosultságokat, hozzáférési szabályokat és titkosítást, amelyek segítenek védeni az adatokat illetéktelen hozzáféréstől.
- **Teljesítmény**: Az Oracle adatbázis rendelkezik optimalizálási funkciókkal, amelyek javítják a teljesítményt és csökkentik az adatbázis kezelési költségeit.
- **Adatintegráció**: Az Oracle adatbázis integrálható más adatforrásokkal, például más adatbázisokkal, alkalmazásokkal és adatfolyamokkal, így egységes adatmodellt hozhat létre.
- **Kiterjedt eszközkészlet**: Az Oracle adatbázis rendelkezik számos eszközzel, amelyek segítenek az adatbázis-kezelésben, például adatbázis-adminisztrációs eszközökkel, adatbányászati megoldásokkal és üzleti intelligencia eszközökkel.



7.3.1.10 Adatbázis-kezelés – SQL Server

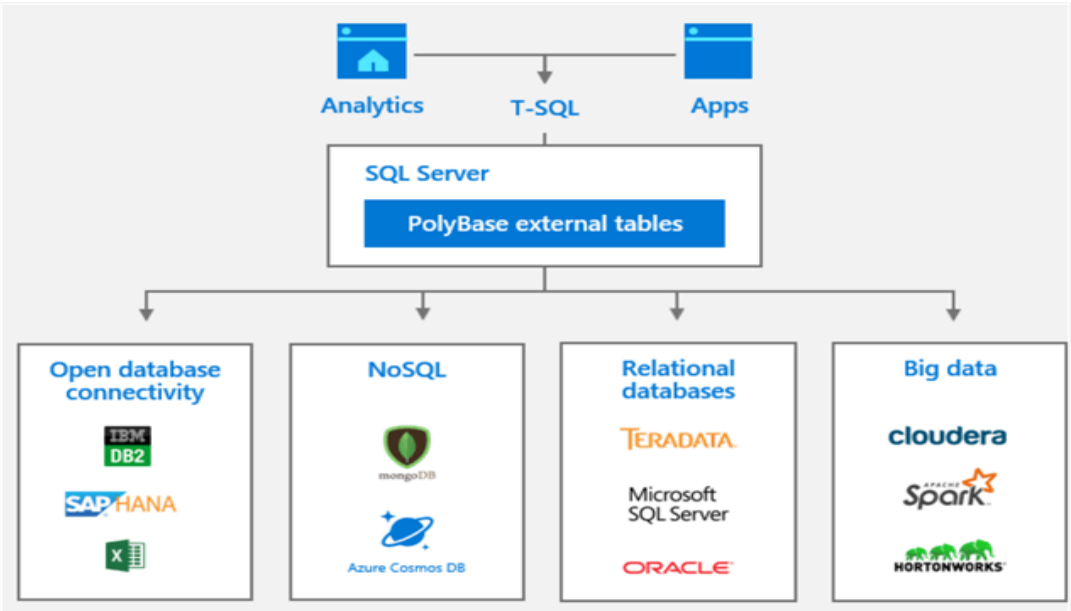
SQL Server egy relációs adatbázis-kezelő szoftver, amelyet a Microsoft fejleszt. Az SQL Server segítségével adatokat tárolhatunk, kezelhetünk és lekérdezhetünk. A szoftver számos funkciót kínál, például tranzakciókezelést, biztonsági funkciókat, replikációt, olvasás-írás hozzáférést, adatfolyamokat és analitikai funkciókat.

A SQL Server támogatja a T-SQL (Transact-SQL) nyelvet, amely a Microsoft által fejlesztett SQL változata. Az adatbázis-kezelő képes tranzakciókat kezelni, ami azt jelenti, hogy egyszerre több műveletet is elvégezhetünk, és ha bármelyik művelet sikertelen, az adatbázis visszaáll az előző állapotára.

A SQL Server támogatja a replikációt is, ami azt jelenti, hogy az adatokat másik számítógépre is át lehet másolni, így biztonsági mentést lehet készíteni az adatokról. Az adatbázis-kezelő rendelkezik olvasás-írás hozzáféréssel is, ami lehetővé teszi a felhasználók számára az adatok olvasását és írását az adatbázisban.

Az SQL Server egy analitikai adatbázis-kezelő is, amely számos eszközt és funkciót kínál az adatok elemzéséhez és riportok készítéséhez. Az adatbázis-kezelő számos analitikai eszközt tartalmaz, mint például az Analysis Services, a Reporting Services és az Integration Services.

Az SQL Server rendelkezik biztonsági funkciókkal is, mint például az adatbázisok és a felhasználók hozzáférésének korlátozása, a biztonsági mentések készítése, az adatbázisok titkosítása és az auditálás.



7.3.1.11 Adatbázis-kezelés – NoSQL

NoSQL (not only SQL) egy olyan adatbázis-kezelési megközelítés, amely a hagyományos relációs adatbázis-kezelési rendszerek alternatívájaként jött létre.

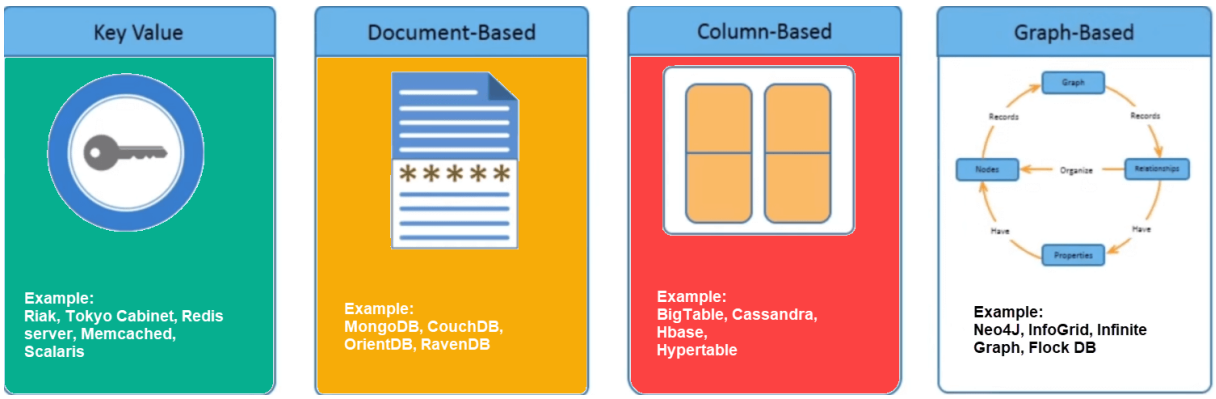
A NoSQL adatbázis-kezelő rendszerek célja, hogy olyan nagy méretű és nagy adatmennyiséggel rendelkező rendszereket tudjanak hatékonyan kezelni, amelyek túlterheltté teszik a hagyományos relációs adatbázisokat.

A NoSQL adatbázisok fő jellemzője, hogy nem használnak táblákat és relációkat, ahogy azt a hagyományos relációs adatbázisokban látjuk. Ehelyett az adatokat dokumentumok, kulcs-érték párok, gráfok vagy oszlop-alapú adatmodellek formájában tárolják.

A NoSQL adatbázisokat széles körben alkalmazzák az internetes alkalmazásokban, ahol a nagy sebesség és skálázhatóság nagyon fontosak.

A NoSQL adatbázisok között számos különböző típus található, például dokumentum alapú adatbázisok, kulcs-érték alapú adatbázisok, gráf adatbázisok, oszlop-alapú adatbázisok és egyéb speciális adatbázisok, mint például idő soros adatbázisok.

Néhány példa a legnépszerűbb NoSQL adatbázis-kezelőkre: MongoDB, Cassandra, Redis, Couchbase, Neo4j, HBase, DynamoDB.

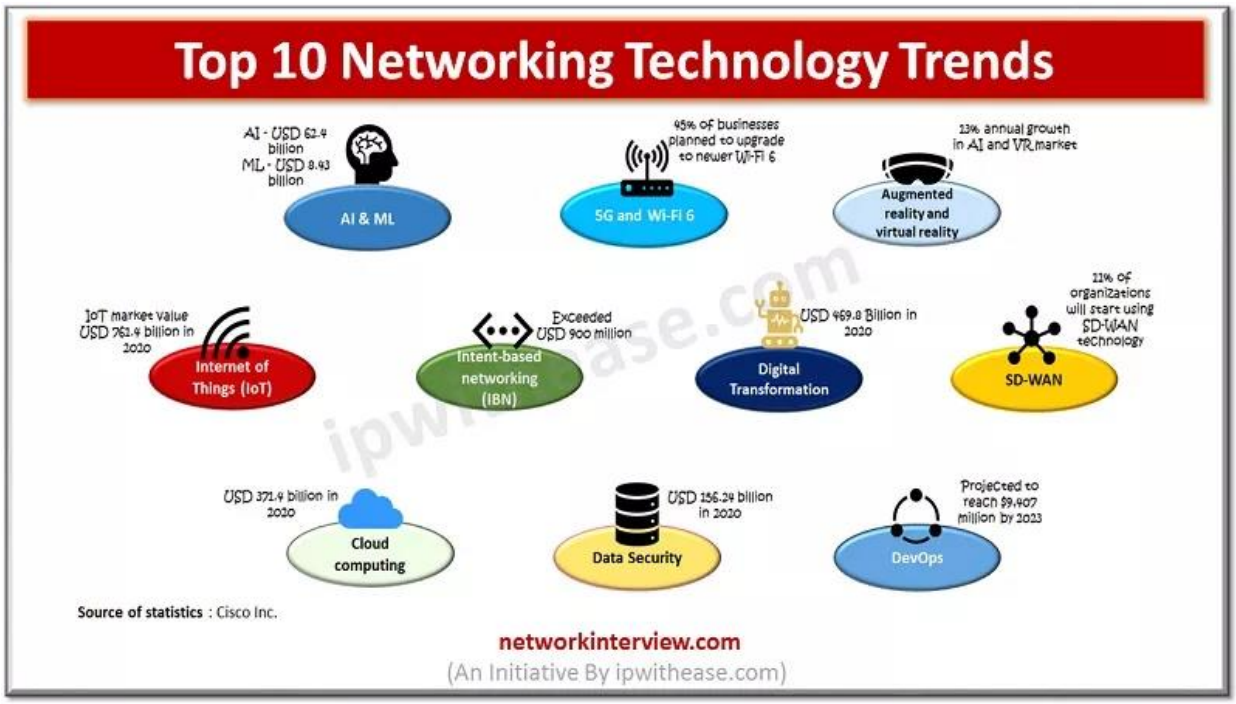


7.4 Hálózati technológiák

A hálózati technológiák fontos szerepet töltenek be a vállalati architektúrában. Ezek a technológiák lehetővé teszik a különböző rendszerek és alkalmazások közötti kommunikációt és integrációt, valamint biztosítják az adatok átvitelének biztonságát és megbízhatóságát.

A vállalati architektúra tervezésénél a hálózati technológiák megválasztása és beállítása fontos feladat. A megfelelő hálózati architektúra lehetővé teszi a rendszerek és alkalmazások optimális működését, és biztosítja a különböző felhasználói eszközök és platformok közötti integrációt.

A hálózati technológiák szerepe tovább nőtt a felhőalapú infrastruktúrák térnyerésével. A felhőalapú szolgáltatások használatával a vállalatok nem csak a saját belső hálózatukon belül tudnak kommunikálni és adatokat továbbítani, hanem lehetőségük van külső felhőszolgáltatásokat is igénybe venni, amelyekkel akár globális szinten is integrálhatják a rendszereiket.



7.4.1 Típus

A hálózati technológia típusok széles skáláját foglalják magukba, és számos különböző kategóriába sorolhatók. Az alábbiakban felsorolok néhányat a leggyakrabban használt típusok közül:

Ethernet: Az Ethernet a legelterjedtebb hálózati technológia, amely az adatokat csomagokban továbbítja az IP hálózaton keresztül. Az Ethernet hálózatok általában a TCP/IP protokollt használják, és lehet vezetékes vagy vezeték nélküli.

Wi-Fi: A Wi-Fi a vezeték nélküli hálózati technológia, amely lehetővé teszi a vezeték nélküli eszközök számára, hogy kapcsolódjanak az internethez vagy más hálózati eszközökhöz. A Wi-Fi hálózatok általában a 802.11 standardot követik.

Bluetooth: A Bluetooth a vezeték nélküli hálózati technológia, amely lehetővé teszi az eszközök közötti adatátvitelt rövid távolságon belül. A Bluetooth hálózatok általában a készülékek közötti adatkapcsolatok kialakítására használják.

MPLS: Az MPLS (Multiprotocol Label Switching) egy hálózati technológia, amely az adatok továbbítását előre meghatározott útvonalakon keresztül végzi, amelyeket az MPLS címkék azonosítanak. Az MPLS hálózatokat gyakran a nagyvállalatok és szolgáltatók használják.

VPN: A VPN (Virtual Private Network) egy biztonságos kommunikációs csatorna, amely lehetővé teszi a távoli felhasználók számára, hogy biztonságosan kapcsolódjanak a vállalati hálózathoz az interneten keresztül. A VPN hálózatokat általában a vállalatok és szervezetek használják a biztonságos távoli hozzáférés megteremtéséhez.

Ezen túlmenően számos más hálózati technológia is létezik, például az ATM (Asynchronous Transfer Mode), a Frame Relay, a SONET (Synchronous Optical Networking) és a FDDI (Fiber Distributed Data Interface), amelyek általában a nagyvállalatok és szolgáltatók számára nyújtanak hálózati megoldásokat.

7.4.2 Protokoll

A hálózati protokollok olyan szabványok és szabályok, amelyeket a hálózati eszközök használnak az adatok továbbítására és a kommunikációra a hálózaton keresztül. A hálózati protokollok meghatározzák az adatok formátumát, az adatok továbbításának módját, az eszközök címzési rendszerét, az adatok biztonságát és az átviteli sebességet.

TCP/IP: Az Internet Protokoll (IP) és a Transmission Control Protocol (TCP) az internetes kommunikáció legelterjedtebb protokolljai. Az IP felelős az adatok továbbításáért, míg a TCP biztosítja az adatok megbízható továbbítását a hálózaton keresztül.

HTTP: Az HTTP (Hypertext Transfer Protocol) a webes kommunikáció szabványa, amely a kliensek és a szerverek közötti adatátvitelt szabályozza. Az HTTP-t gyakran a weboldalak letöltéséhez és a böngészők használatához használják.

FTP: Az FTP (File Transfer Protocol) a fájlok átvitelének protokollja a hálózaton keresztül. Az FTP-t gyakran a fájlok feltöltéséhez és letöltéséhez használják.

DNS: Az DNS (Domain Name System) az interneten használt domain nevek és IP-címek megfeleltetésének rendszere. Az DNS segít az internetes eszközöknek azonosítani az internetes tartalom helyét és elérhetőségét.

SMTP: Az SMTP (Simple Mail Transfer Protocol) az email továbbításának protokollja. Az SMTP segít az email szervereknek az email továbbítását az interneten keresztül.

Ezen kívül számos más hálózati protokoll is létezik, amelyek az adott hálózati funkcióktól és céloktól függenek. A hálózati protokollok fontos szerepet játszanak a hálózati kommunikációban és a hálózati biztonságban is.

7.4.3 Topológia

A hálózati topológia a hálózat geometriai elrendezésének leírására szolgáló fogalom. A hálózati topológia a hálózat fizikai és logikai összeköttetéseinek megadásával írja le, hogy az adatok és kommunikáció miként áramlanak a hálózaton keresztül.

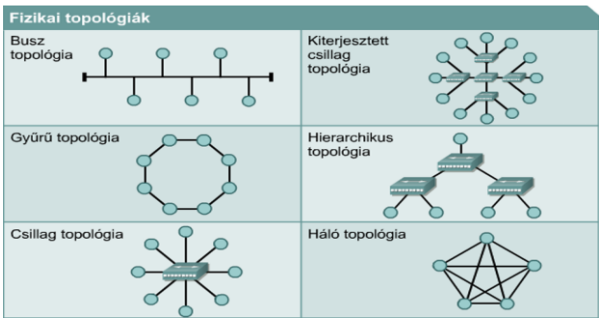
A hálózati topológia három fő típusa

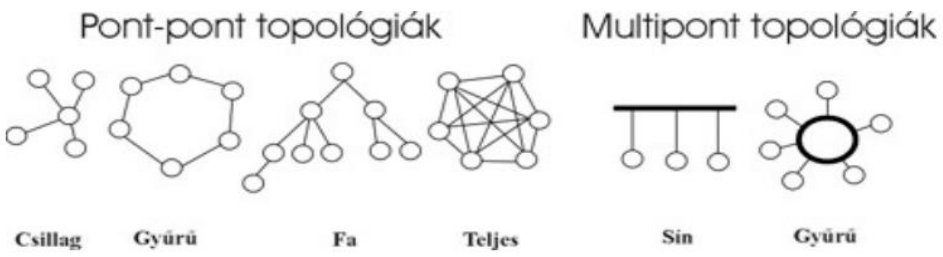
Csillag topológia: Ez a leggyakoribb topológiai forma, amelyben minden eszköz közvetlenül kapcsolódik a központi eszközhöz, például egy hubhoz vagy switchhez. A csillag topológia könnyen telepíthető és karbantartható, azonban, ha a központi eszköz meghibásodik, az egész hálózat leállhat.

Busz topológia: A busz topológia esetében az eszközök egy közös kommunikációs vonalra csatlakoznak. Az adatok minden eszközön áthaladnak, és csak azok az eszközök tudják fogadni az adatokat, amelyek a vonalon vannak. A busz topológia egyszerű, de ha a közös vonalon probléma lép fel, az egész hálózat megbénulhat.

Gyűrű topológia: A gyűrű topológia esetében az eszközök egy zárt hurkot alkotnak, amelyen keresztül az adatok körbe-körbe haladnak. A gyűrű topológia megbízható, mivel, ha egy csomópont meghibásodik, az adatok továbbra is továbbítódnak a hálózaton.

A hálózati topológia kiválasztása az adott hálózat igényeitől, méretétől és az eszközök elhelyezkedésétől függ.





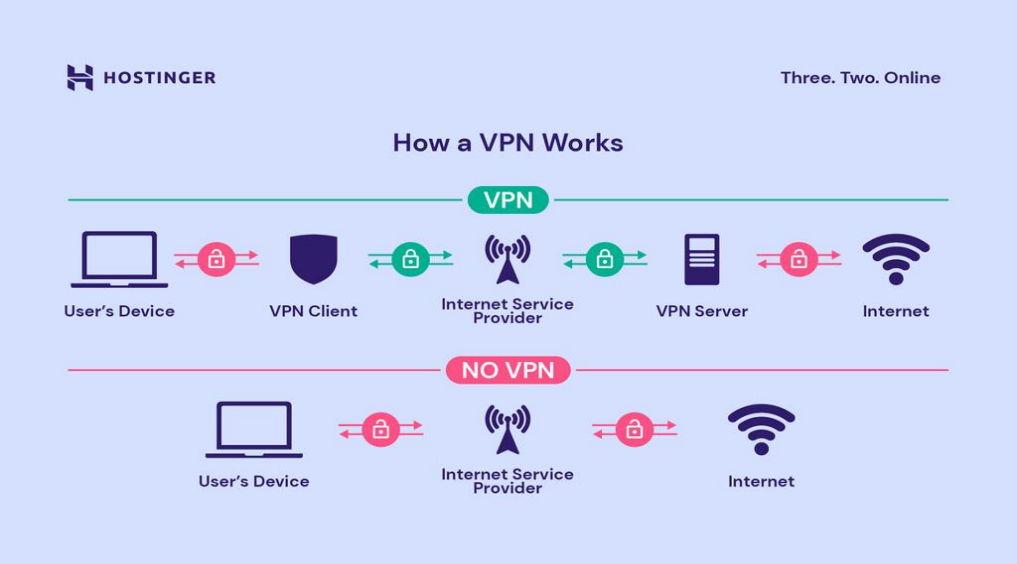
7.4.4 Hálózati technológiák - VPN

A VPN (Virtual Private Network) egy biztonságos és titkosított kommunikációs csatorna, amely lehetővé teszi a felhasználók számára, hogy biztonságosan kapcsolódjanak hozzáférési pontokhoz a hálózaton keresztül. A VPN általában az interneten keresztül hozzáférhető, és titkosítja az adatokat, így a felhasználók biztonságosan küldhetik és fogadhatják az adatokat a hálózaton keresztül.

A VPN használatakor az adatokat a VPN szerveren keresztül továbbítják, amely titkosítja az adatokat, és csak a megfelelő kulccsal rendelkező felhasználók tudják megfejteni az adatokat. A VPN lehetővé teszi a felhasználók számára, hogy biztonságosan kapcsolódjanak hozzáférési pontokhoz, például az irodai hálózathoz, vagy elérjék az internetet olyan helyekről, amelyek nem biztonságosak.

A VPN-t gyakran használják a vállalatok és a szervezetek azon dolgozói számára, akik távoli helyen dolgoznak vagy utaznak, és biztonságosan kell hozzáférniük a vállalati hálózathoz vagy erőforrásokhoz. A VPN használata továbbá lehetővé teszi a felhasználók számára, hogy elkerüljék az internetes korlátozásokat, például az internetes cenzúrát, valamint, hogy megvédjék magukat a hackerek és az online adathalászok támadásaitól.

A VPN általában különböző technológiákon keresztül működhet, például az IPsec (Internet Protocol Security), a SSL/TLS (Secure Sockets Layer/Transport Layer Security) vagy a PPTP (Point-to-Point Tunneling Protocol) protokollok használatával.



7.5 Hardware technológiák

7.5.1 Méretezés

A hardware méretezés célja az, hogy az adott alkalmazáshoz szükséges erőforrásokat megfelelően allokálják annak érdekében, hogy a rendszer hatékonyan és megbízhatóan működjön.

Hardware méretezési technikák

Skálázhatóság: A rendszer skálázható legyen, azaz képesnek kell lennie az erőforrások hozzáadására vagy eltávolítására a rendszer terhelésének növekedése vagy csökkenése szerint.

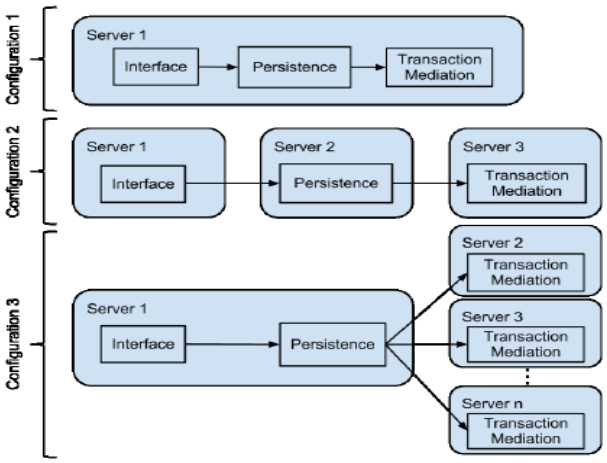
Teljesítmény elemzés: A rendszer teljesítményét rendszeresen mérni kell, hogy azonosítsák a gyenge pontokat, és a rendszer javítására lehetőséget találjanak.

Terhelés tesztelés: A rendszer terhelés tesztelése segít meghatározni, hogy milyen mennyiségű adatot képes a rendszer kezelni anélkül, hogy teljesítményproblémák merülnének fel.

Alkalmazás függőségek kezelése: Az alkalmazások függőségeinek kezelése kulcsfontosságú a megbízható rendszer fenntartásához. A rendszernek képesnek kell lennie az alkalmazások függőségeinek azonosítására és azok kezelésére.

Biztonsági mentés: A rendszernek biztosítania kell az adatok biztonsági mentését, hogy a rendszer katasztrófa esetén is helyreállítható legyen.

Vészterv: Fontos, hogy a rendszer üzemeltetői rendelkezzenek vésztervvel, amelynek célja a rendszer helyreállítása katasztrófa esetén.



7.5.2 Teljesítménymérés

Hardware teljesítménymérés eszközei és módszerei

Teljesítménymérő szoftverek: Az ilyen típusú szoftverek általában tesztkörnyezetet biztosítanak a számítógépek teljesítménymméréséhez és elemzéséhez. Például az AIDA64 és a PassMark PerformanceTest.

Hardver monitorok: Ezek az eszközök lehetővé teszik a hardver elemek teljesítményének valós idejű figyelését. Például a CPU-Z és a GPU-Z.

Terhelésmérő szoftverek: Ezek a szoftverek segítenek meghatározni, hogy milyen terhelésre képes egy adott hardverkomponens. Például az Intel XTU és az AMD OverDrive.

Profilozó szoftverek: Ezek az eszközök lehetővé teszik az alkalmazások futásidejű teljesítményének elemzését. Például az Intel VTune és a AMD CodeXL.

Hálózati monitorok: Az ilyen típusú eszközök lehetővé teszik a hálózati forgalom figyelését és elemzését. Például a Wireshark és a Microsoft Network Monitor.

Ezen eszközök segítségével a hardver teljesítményét mérhetjük és elemezhetjük, és lehetőséget kapunk a hatékonyabb és optimalizáltabb hardver kialakítására és használatára.

7.5.3 Teljesítmény elemzés

Hardware teljesítmény elemzése feladatai

Teljesítményigény meghatározása: A rendszer tervezésekor fontos meghatározni, hogy az alkalmazásnak mekkora teljesítményre van szüksége. Ezt az igényt figyelembe véve lehet a megfelelő hardvert kiválasztani.

Teljesítmény elemzése: A rendszer teljesítményének elemzése során mérhetők az egyes hardverkomponensek teljesítményének jellemzői. Az elemzés során vizsgálják a hardver CPU, memória és tárolókapacitását, valamint a hálózati sávszélességet.

Teljesítményoptimalizálás: Az elemzések alapján lehetőség nyílik a rendszer teljesítményének optimalizálására. Az optimalizálás során javíthatók a hardverkomponensek kapacitásai, vagy a rendszer konfigurációja. Az optimalizálásnak célja, hogy a rendszer a lehető legnagyobb hatékonysággal működjön.

Terhelés tesztelés: A rendszer terhelés tesztelése során mérhető a rendszer által kezelt tranzakciók száma és sebessége. A tesztelés során vizsgálják a rendszer stabilitását és teljesítményét nagy terhelés mellett is.

Megbízhatósági tesztelés: A rendszer megbízhatóságát a megbízhatósági tesztek során vizsgálják. A tesztek célja, hogy a rendszer a lehető legmagasabb rendelkezésre állással és minimális leállással működjön. A megbízhatósági tesztek során vizsgálják a rendszer redundanciáját, illetve a mentési és helyreállítási lehetőségeket.

7.5.4 Nagy adatbázisok hardware méretezése

Nagy adatbázisok esetén a hardware méretezése kulcsfontosságú.

Processzor: Nagy adatbázisokhoz több processzor szükséges, amelyek lehetnek egyetlen nagy processzor vagy több kisebb processzor, amelyek párhuzamosan dolgoznak. A processzorok száma a terhelés és a teljesítmény igényei alapján választható.

Memória: A nagy adatbázisok számára nagy mennyiségű memória szükséges az adatok hatékony kezeléséhez. Az adatbázis méretének és a használat módjának függvényében a memóriaméret megválasztása fontos szerepet játszik a teljesítményben.

Tárolóeszközök: A nagy adatbázisokhoz szükség van nagy mennyiségű tárolókapacitásra, amely lehet merevlemez, szalagos adattároló vagy akár felhő alapú adattárolás. Az adattárolók kapacitásának függvényében a nagy adatbázisoknak több tárolóeszközre lehet szükségük.

Hálózati infrastruktúra: A nagy adatbázisokhoz szükség van gyors hálózati kapcsolatra, amely lehet Ethernet, InfiniBand vagy akár fiber optic. A hálózati infrastruktúrának lehetővé kell tennie az adatok gyors átvitelét és az adatbázis szerverek közötti kommunikációt.

Biztonsági eszközök: A nagy adatbázisok védelme érdekében különféle biztonsági eszközökre lehet szükség, például tűzfalakra, vírusvédelmi szoftverekre és az adatok biztonsági másolatok készítésére.

Skálázhatóság: A nagy adatbázisok esetében fontos, hogy a hardver könnyen skálázható legyen, amely lehetővé teszi az adatbázis méretének növelését és a terhelés növekedésére való reagálást. Ezt meg lehet valósítani például hozzáadott processzorokkal vagy memóriával, valamint további tárolókapacitással.

Az adatbázisok hardware eszközei a számítógépes rendszerek, amelyek az adatbázisokat futtatják. Ide tartoznak a szerverek, amelyeknek magas processzor teljesítményük van, nagy mennyiségű memóriájuk van, valamint megbízható és biztonságos adattárolókat használnak. Az adatbázisok számára fontos a nagy tárolási kapacitás, így az adattárolók (merevlemezek vagy SSD-k) is fontosak az adatbázisok teljesítménye szempontjából. Az adatbázisokat futtató szervereknek gyakran van több hálózati kapcsolata is, amelyek az adatok továbbítására és a hálózati kommunikációra szolgálnak. Az adatbázisok teljesítményének optimalizálása érdekében a szerverekben találhatók hardveres gyorsítók, mint például a grafikus processzorok (GPU-k) vagy a TPU-k (Tensor Processing Units). A nagy adatbázisok esetén az adatok kezeléséhez és

feldolgozásához több szerverre van szükség, amelyek összekapcsolva alkotják az adatbázis rendszert. Ezeket a szervereket gyakran úgynevezett adatközpontokban helyezik el, ahol speciális infrastruktúra és klímafeltételek biztosítják az optimális működést.

7.5.5 Felhő (Cloud) alapú megoldások

Felhő alapú megoldások hardver lehetőségei

Virtuális gépek: A virtuális gépek lehetővé teszik az alkalmazások és az adatok független futtatását a hardver erőforrások megosztásával.

Több processzor és több mag: A több processzor és több mag lehetővé teszi a felhő alapú alkalmazások hatékonyabb feldolgozását és a nagy adatmennyiségek gyorsabb kezelését.

Nagy tárolókapacitás: A felhő alapú megoldások nagy tárolókapacitást kínálnak a nagy adatmennyiségek tárolásához és kezeléséhez.

Magas sávszélességű hálózati kapcsolatok: A magas sávszélességű hálózati kapcsolatok lehetővé teszik az adatok gyors átvitelét a felhő alapú megoldások között, így hatékonyabbá teszik az alkalmazásokat és az adatokat.

Automatikus skálázás: A felhő alapú megoldások automatikus skálázási lehetőségekkel rendelkeznek, amelyek lehetővé teszik a rendszer kapacitásának dinamikus növelését a megnövekedett terhelés kezeléséhez.

Rugalmasság: A felhő alapú megoldások rugalmasságot biztosítanak a felhasználók számára, hogy az erőforrásokat az igényeknek megfelelően használhassák és az alkalmazásokat a saját igényeik szerint alakítsák ki.

Rendszerintegráció: A felhő alapú megoldások lehetővé teszik az alkalmazások és az adatok integrációját más rendszerekkel, így egyszerűbbé teszik a rendszerek közötti kommunikációt és adatátvitelt.

Biztonság: A felhő alapú megoldások biztonságos környezetet biztosítanak az alkalmazások és az adatok számára, biztosítják a megfelelő biztonsági mentési lehetőségeket és redundanciát az adatvesztés elkerülése érdekében, valamint a felhasználói adatok védelmét is.

Az állandó fejlesztéseknek köszönhetően a felhő alapú megoldások hardver lehetőségei egyre hatékonyabbak lesznek, amely lehetővé teszi a nagyobb adatmennyiségek hatékonyabb kezelését és feldolgozását.

7.5.6 Felhő (Cloud) szolgáltatások

A felhőszolgáltatások olyan szolgáltatások, amelyek lehetővé teszik az alkalmazások és az adatok tárolását és feldolgozását a felhőben, azaz egy központi szerverhálózaton keresztül.

Felhőszolgáltatások előnyei

Rugalmasság: A felhőszolgáltatások lehetővé teszik az erőforrások könnyű bővítését és csökkentését az alkalmazkodás érdekében a változó terheléshez.

Hatékonyság: A felhőszolgáltatások lehetővé teszik az alkalmazások hatékony feldolgozását és a nagy adatmennyiségek kezelését.

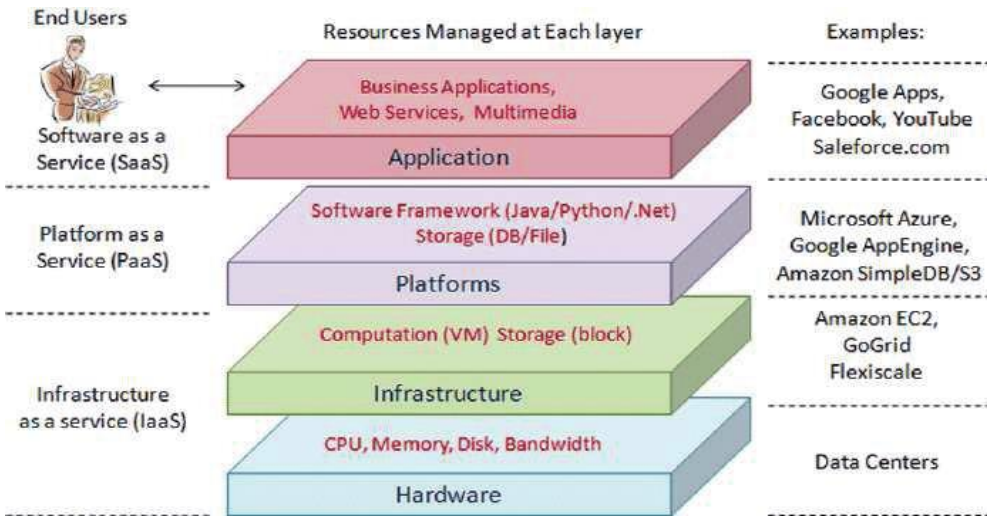
Alacsonyabb költségek: A felhőszolgáltatások általában alacsonyabb költséggel járnak, mint a helyi infrastruktúra építése és üzemeltetése.

Gyorsabb elérhetőség: A felhőszolgáltatások lehetővé teszik az alkalmazások és az adatok gyorsabb elérhetőségét a felhasználók számára.

Biztonság: A felhőszolgáltatások biztonságos környezetet biztosítanak az adatok és az alkalmazások számára.

Rendszerintegráció: A felhőszolgáltatások lehetővé teszik az alkalmazások és az adatok integrációját más rendszerekkel.

Automatizált folyamatok: A felhőszolgáltatások automatizált folyamatokat biztosítanak az alkalmazások és az adatok kezeléséhez, amelyek csökkentik az emberi beavatkozások szükségességét.



7.5.6.1 Felhő (Cloud) szolgáltatások típus

A felhőszolgáltatások több típusra oszthatók attól függően, hogy milyen szintű erőforrásokat és szolgáltatásokat kínálnak a felhasználók számára. A legfontosabb típusok a következők:

Infrastruktúra-szolgáltatások (IaaS): Az infrastruktúra-szolgáltatások lehetővé teszik az IT-infrastruktúra erőforrásainak használatát, például a szervereket, a hálózatokat, a tárolóeszközöket és a virtualizációs technológiákat.

Platform-szolgáltatások (PaaS): A platform-szolgáltatások a szoftverfejlesztési platformokat kínálják, amelyek segítségével a felhasználók létrehozhatnak és futtathatnak saját alkalmazásokat a felhőben.

Szoftver-szolgáltatások (SaaS): A szoftver-szolgáltatások a felhasználóknak kész alkalmazásokat kínálnak, amelyeket a felhőben lehet futtatni és használni, például az e-mail szolgáltatások, a CRM rendszerek, az üzleti intelligencia eszközök stb.

Függőben lévő szolgáltatások (DaaS): A függőben lévő szolgáltatások lehetővé teszik a felhasználók számára, hogy egy adott szolgáltatást hívjanak meg a felhőben, például a nyomtatást, a távoli hozzáférést stb.

Biztonsági szolgáltatások (SECaaS): A biztonsági szolgáltatások a felhőben kínálnak biztonsági megoldásokat, például tűzfalakat, vírusirtókat, beléptető rendszereket stb.

Adat-tároló szolgáltatások (STaaS): Az adattárolási szolgáltatások lehetővé teszik az adatok tárolását a felhőben, például a fájlok, adatbázisok, archiválási rendszerek stb.

Ezek az alapvető felhőszolgáltatások lehetővé teszik a felhasználók számára az IT-infrastruktúra erőforrásainak, az alkalmazásoknak és az adatoknak a használatát a felhőben, ami a költséghatékony, rugalmas és hatékony munkavégzést teszi lehetővé a felhasználók számára.

7.6 Biztonsági technológiák

Biztonsági technológiák az adatok és az információk védelmét szolgálják az illetéktelen hozzáférés, a káros szoftverek, a lopás, a csalás és más veszélyek ellen.

7.6.1 Hálózatbiztonsági eljárások

A hálózati biztonsági eljárások olyan módszerek és technikák, amelyeket a hálózat biztonságának javítása érdekében alkalmaznak. Ezek az eljárások segítenek megvédeni a hálózatot a biztonsági fenyegetésektől, például a vírusoktól, a rosszindulatú szoftverektől, az illetéktelen hozzáféréstől és az adatlopástól.

Eljárások és módszerek

Hálózati monitorozás: A hálózati monitorozás segítségével az IT szakemberek folyamatosan figyelemmel kísérik a hálózatot és azonosítják az esetleges biztonsági problémákat. Az ilyen monitorozó eszközök lehetővé teszik a hálózati forgalom elemzését és az anomáliák, például a támadások azonosítását.

Tűzfalak: A tűzfalak olyan biztonsági megoldások, amelyek segítenek megvédeni a hálózatot az illetéktelen hozzáférésektől és a támadásoktól. A tűzfalak általában beállított szabályok szerint döntenek arról, hogy milyen forgalmat engednek át a hálózaton, és milyen forgalmat blokkolnak.

Vírusvédelem: A vírusvédelem fontos szerepet játszik a hálózati biztonságban. Az antivírus programok képesek azonosítani és eltávolítani a kártékony szoftvereket a hálózaton.

Hálózati hozzáférés-kezelés: A hálózati hozzáférés-kezelés azonosítja, hogy mely felhasználók jogosultak a hálózati erőforrások elérésére, és milyen mértékben. Ez segít megakadályozni az illetéktelen hozzáféréseket a hálózatra.

Hálózati sérülékenység felmérés: A hálózati sérülékenység felmérés során az IT szakemberek azonosítják a hálózat gyenge pontjait, amelyeket a támadók kihasználhatnak, majd javaslatot tesznek a sérülékenységek kiküszöbölésére.

Jelszókezelés: A jelszókezelés segít megvédeni a hálózatot a nem kívánt hozzáférésektől. A felhasználóknak erős és összetett jelszavakat kell használniuk, és rendszeresen kell módosítaniuk azokat.

Adatmentés: Az adatmentés az adatok biztonsági mentése, amely segít megőrizni a fontos adatokat és visszaállítani azokat

VPN: A VPN (Virtual Private Network) biztonságos csatornát biztosít az interneten keresztül történő adatátvitelhez. Ez a hálózati biztonsági eljárás lehetővé teszi a távoli hozzáférést a hálózathoz, miközben védelmet nyújt a külső fenyegetésekkel szemben.

Riasztási rendszerek: A riasztási rendszerek érzékelik a biztonsági fenyegetéseket, és értesítik az illetékes személyeket, hogy sürgősen intézkedni tudjanak.

Hálózati kódolás: A hálózati kódolás az adatok titkosítását jelenti, hogy azokat csak azok a személyek láthassák, akik jogosultak az adott adatok megtekintésére.

Ezek az eljárások csak néhány példa a hálózati biztonsági eljárások közül, amelyeket a hálózat védelmére alkalmazhatnak. A biztonsági szakemberek folyamatosan monitorozzák a hálózatot.

7.6.2 Hálózatbiztonsági protokollok

A hálózat biztonsági protokollok olyan szabványok és eljárások, amelyek segítenek megvédeni a hálózatot a biztonsági fenyegetésektől, például a vírusoktól, a rosszindulatú szoftverektől, a támadásoktól és az adatlopástól. Ezek a protokollok szabályozzák a hálózati forgalmat, és lehetővé teszik a hálózati eszközök és alkalmazások közötti biztonságos kommunikációt.

Hálózati biztonsági protokollok

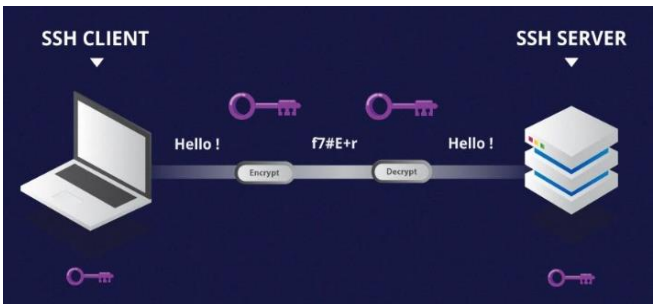
SSL/TLS: Az SSL (Secure Sockets Layer) és a TLS (Transport Layer Security) protokollok titkosítják a kommunikációt a két végpont között. Ez a protokoll általában weboldalakon és e-mail szervereken alkalmazható, hogy megvédjék a kommunikációt a külső fenyegetésektől.

IPsec: Az IPsec (Internet Protocol Security) protokoll az interneten keresztüli adatátvitel biztonságosabbá tételére szolgál. A hálózati forgalom titkosítását és azonosítását biztosítja a virtuális magánhálózatok (VPN) esetében.

SSH: Az SSH (Secure Shell) protokoll egy biztonságos kommunikációs protokoll, amely lehetővé teszi a biztonságos távoli hozzáférést. Az SSH titkosítja a kommunikációt, és így védi a hálózatot a külső fenyegetésektől.

SNMPv3: Az SNMPv3 (Simple Network Management Protocol version 3) protokoll biztonságosabbá teszi a hálózatok menedzsmentjét és ellenőrzését. Az SNMPv3 lehetővé teszi a hálózati eszközök biztonságos konfigurálását és azok állapotának ellenőrzését. Az SNMPv3 titkosítja az adatokat, és így védi azokat a külső fenyegetésektől.

RADIUS: A RADIUS (Remote Authentication Dial-In User Service) protokoll lehetővé teszi a hálózati hozzáférés-ellenőrzést, a felhasználók azonosítását és hitelesítését. A RADIUS biztonságos kommunikációs csatornát biztosít, és megakadályozza az illetéktelen hozzáféréseket a hálózathoz, és biztonságos hozzáférést biztosít az engedélyezett felhasználók számára.



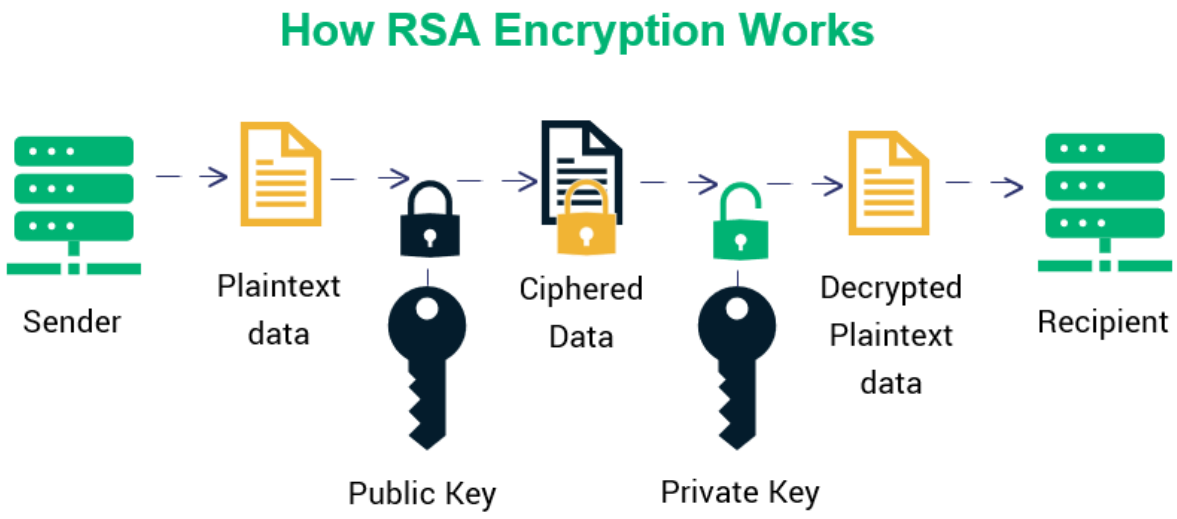
7.6.3 Hálózat titkosítási technológiák

A számítógépes hálózat titkosításának két fő technológiája a szimmetrikus és az aszimmetrikus titkosítás.

A szimmetrikus titkosítás során a titkosított adatok és az azt visszafejtő kulcs ugyanaz, így ezt a módszert egyszerűbb implementálni és gyorsabb is lehet, mint az aszimmetrikus titkosítást. Az adatok és a kulcs azonban megosztott, ezért a kulcsbiztonság fontos tényező, és a kulcsokat időnként meg kell változtatni.

Az aszimmetrikus titkosításnál két különböző kulcsot használnak: egyet a titkosításhoz és egyet a visszafejtéshez. Ez lehetővé teszi, hogy az egyik kulcs nyilvános legyen, míg a másikat csak a titkosított adatok visszafejtéséhez használják. Az aszimmetrikus titkosítás általában lassabb, mint a szimmetrikus titkosítás, de nagyobb biztonságot nyújt, mivel a nyilvános kulcs nem árthat a titkosításnak. Az egyik leggyakoribb aszimmetrikus titkosítási algoritmus a RSA.

További technológiák közé tartoznak az SSL / TLS (Secure Sockets Layer / Transport Layer Security) protokollok, amelyek kriptográfiát és digitális tanúsítványokat használnak a kommunikáció biztonságossá tételéhez, valamint a VPN (Virtual Private Network) technológiák, amelyek biztonságos kapcsolatot biztosítanak két vagy több hálózat között, például az interneten keresztül.



7.6.4 Biztonságos Internet protokollok

A biztonságos internetprotokollok olyan kommunikációs protokollok, amelyek biztonságosabbá teszik az interneten történő adatátvitelt. Ezek a protokollok különféle biztonsági mechanizmusokat alkalmaznak, például az adatok titkosítását, az authentikációt és a hitelesítést.

Biztonságos Internet protokollok

HTTPS: Az HTTPS (HyperText Transfer Protocol Secure) a HTTP biztonságos verziója, amely titkosított kommunikációt biztosít a weboldalak és a felhasználók között.

SSL/TLS: Az SSL (Secure Sockets Layer) és a TLS (Transport Layer Security) protokollok titkosítják a kommunikációt a két végpont között. Ez a protokoll általában weboldalakon és e-mail szervereken alkalmazható, hogy megvédjék a kommunikációt a külső fenyegetésektől.

SSH: Az SSH (Secure Shell) protokoll lehetővé teszi a távoli hozzáférést a hálózathoz, biztonságos módon. Az SSH titkosítja a kommunikációt a távoli számítógépek között, és megakadályozza a külső fenyegetéseket.

IPsec: Az IPsec (Internet Protocol Security) protokoll a hálózati forgalom titkosítását és azonosítását biztosítja a virtuális magánhálózatok (VPN) esetében. Az IPsec az interneten keresztül történő adatátvitelt biztonságosabbá teszi a hálózatban.

DNSSEC: A DNSSEC (Domain Name System Security Extensions) protokoll lehetővé teszi a domain név szerverek adatainak titkosítását és hitelesítését. Ez megakadályozza a DNS támadásokat, amelyek során az adatokat megváltoztatják vagy hamisítják.

Ezek csak néhány példa a biztonságos internetprotokollok közül, amelyeket a kommunikáció biztonságosabbá tételére használhatunk. A protokollok szerepe, hogy megóvják az adatokat az illetéktelen hozzáféréstől, a hacker támadásoktól és más külső fenyegetésektől.

7.6.5 Biztonsági technológiák - Adatbiztonság

Az adatbiztonsági technológiák számos területen működhetnek együtt, hogy biztosítsák az adatok védelmét.

A legfontosabb biztonsági technológiák

Tűzfalak: A tűzfalak a számítógépes hálózatok első védelmi vonalát jelentik. Megakadályozzák a jogosulatlan hozzáférést a hálózatra és az adatokhoz, valamint figyelmeztetést küldenek a biztonsági szakembereknek, ha valamilyen fenyegetés merül fel.

Vírusvédelmi szoftverek: A vírusvédelmi szoftverek célja, hogy megakadályozzák a számítógépekbe és a hálózatokba való vírusos fertőzéseket. Ez a szoftver rendszeresen ellenőrzi a rendszert és az adatokat, hogy megakadályozza a vírusok terjedését.

Titkosítás: A titkosítás adataink védelmére szolgál, amely lehetővé teszi, hogy az adatokat csak a kijelölt címzett olvassa el. Az adatok titkosítása kulcsfontosságú a személyes adatok védelme érdekében, amikor azokat az interneten keresztül küldjük.

Hozzáférés-ellenőrzés: A hozzáférés-ellenőrzés biztosítja, hogy csak a megfelelő személyek férjenek hozzá a számítógép- és hálózati rendszerekhez, adatokhoz és alkalmazásokhoz. Ez a védelmi réteg biztosítja a rendszerek védelmét az illetéktelen hozzáférés ellen.

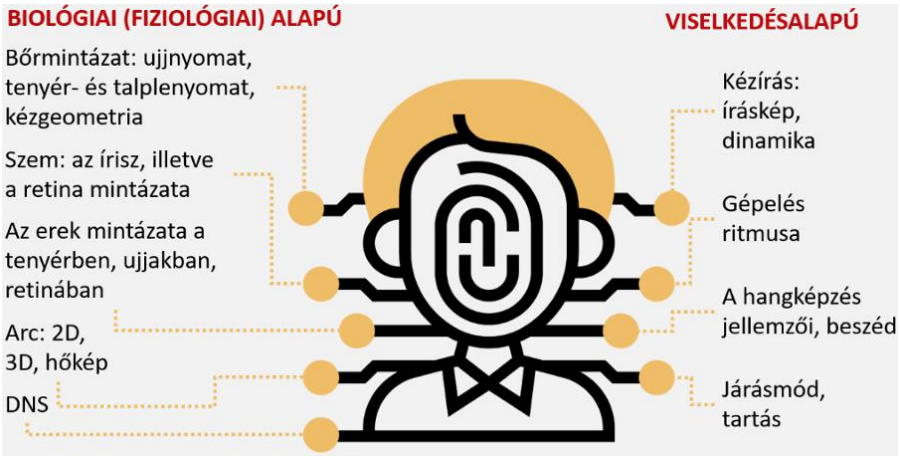
Eseménynaplók: Az eseménynaplók olyan rendszerek, amelyek figyelik és rögzítik az összes rendszereseményt, amelyet a számítógép- és hálózati rendszerekben generál. Ezek az eseménynaplók segítenek a rendszergazdáknak az illetéktelen hozzáférések és a biztonsági incidensek azonosításában és reagálásában.

Jelszóvédelem: A jelszavak biztosítják a felhasználók számára az adatokhoz való hozzáférést. Az erős jelszavak használata fontos az adatok védelme szempontjából. A jelszavakat rendszeresen meg kell változtatni és biztonságos helyen kell tárolni.

Adatmentés: Az adatmentés biztosítja, hogy az adatok biztonságban legyenek, ha azokat véletlenül vagy szándékosan törlik, illetve, ha a rendszer vagy a számítógép meghibásodik. Az adatokat rendszeresen kell menteni és biztonságos helyen kell tárolni.

Kriptográfiai technológiák: A kriptográfiai technológiák lehetővé teszik az adatok titkosítását és biztonságos tárolását. A titkosított adatokat csak azok a felhasználók érhetik el, akiknek megvan a megfelelő hozzáférési joguk.

Biometrikus azonosítók: A biometrikus azonosítók, például az ujjlenyomatok és az arc felismerése, lehetővé teszik a felhasználók számára, hogy azonosítsák magukat, és megakadályozzák a jogosulatlan hozzáférést az adatokhoz.



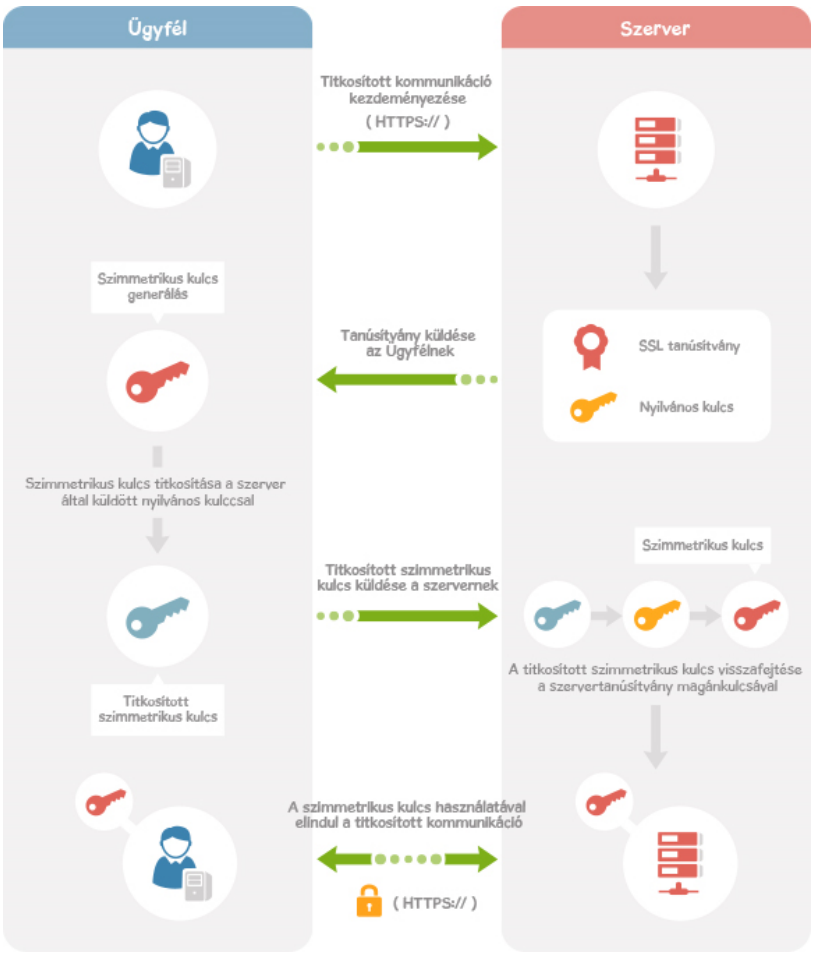
7.6.6 Biztonsági Technológiák – Titkosítás – SSL

Az SSL kapcsolat kiépülése során a kliens HTTPS:// előtag használatával kérést intéz a szerver felé a biztonságos kommunikáció kiépítésének érdekében. Az SSL tanúsítványt használó szerver a kérésre válaszként átküldi az SSL tanúsítványt, és a hozzá tartozó nyilvános kulcsot.

Annak érdekében, hogy a nyilvános kulcshoz tartozó tanúsítványt a kliens érvényesnek lássa, feltétlen megbízható hitelesítés-szolgáltatótól kell származnia. Ilyen nemzetközi hitelesítés-szolgáltató a NETLOCK is.

A kliens az átküldött nyilvános kulcs segítségével titkosítja az automatán generálódott szimmetrikus kulcsot, amellyel majd a titkosított kapcsolat adatáramlása megvalósul. A kliens a szerver SSL tanúsítványához tartozó nyilvános kulccsal titkosított szimmetrikus kulcsát átküldi a szerver részére.

A szerver a magánkulcsával dekódolja a titkosított szimmetrikus kulcsot, így már mindkét kommunikáló fél rendelkezik a szimmetrikus kulccsal, tehát megkezdődhet a szimmetrikus kulccsal titkosított adatok védett, biztonságos áramlása.

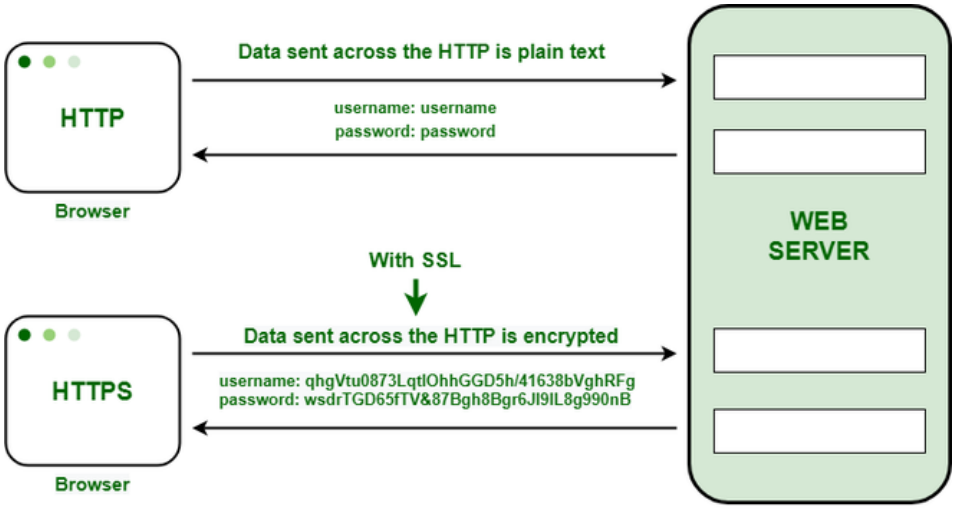


7.6.7 Biztonsági technológiák – Hálózat biztonság - HTTPS

A HTTPS (HyperText Transfer Protocol Secure) az HTTP biztonságosabb verziója, amely titkosított kommunikációt biztosít a weboldalak és a felhasználók között. Az HTTPS használatakor a kommunikáció során a küldött adatok titkosítva vannak, így azokat csak a két végpont (a kliens és a szerver) tudják megfejteni.

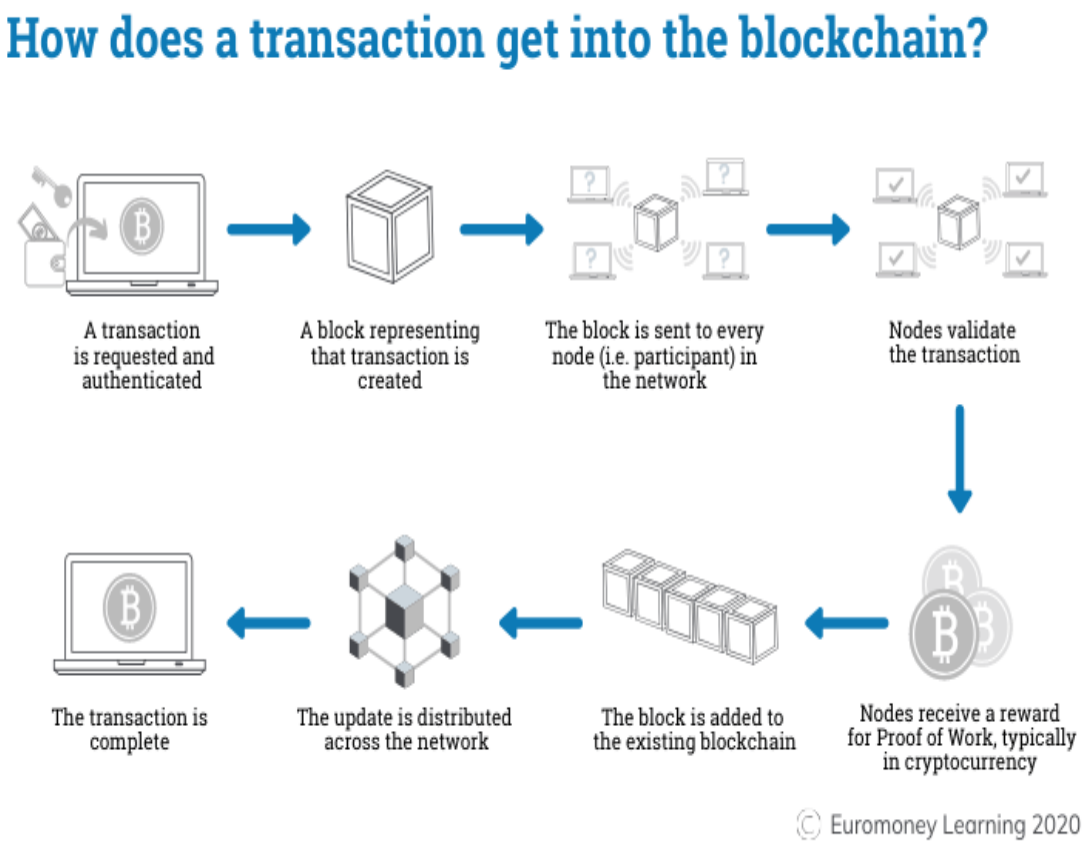
Az HTTPS használatához az adatokat titkosító SSL/TLS (Secure Sockets Layer/Transport Layer Security) protokollra van szükség. Az SSL/TLS protokoll a kommunikációs csatornát titkosítja, és megvédi az adatokat a külső fenyegetésektől, például a hacker támadásoktól és a rosszindulatú programoktól.

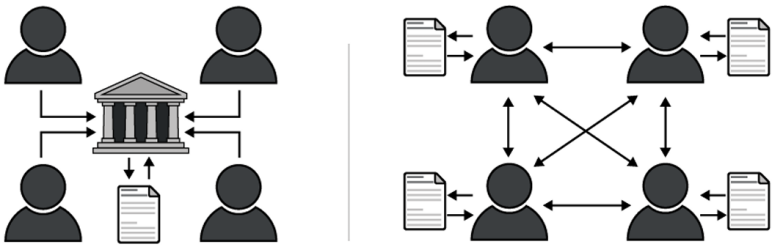
Az HTTPS protokoll biztonságosabbá teszi az internetes kommunikációt, és megvédi a felhasználók személyes adatait, például a jelszavakat, a banki információkat, az e-mail címeket és más személyes adatokat. A weboldalak tulajdonosai is használják az HTTPS protokollt, hogy megóvják a weboldalukat az illetéktelen hozzáféréstől és a hackertámadásoktól, és hogy javítsák a weboldaluk SEO (Search Engine Optimization) értékét a Google keresési rangsorában.



7.6.8 Biztonsági technológiák – Blockchain

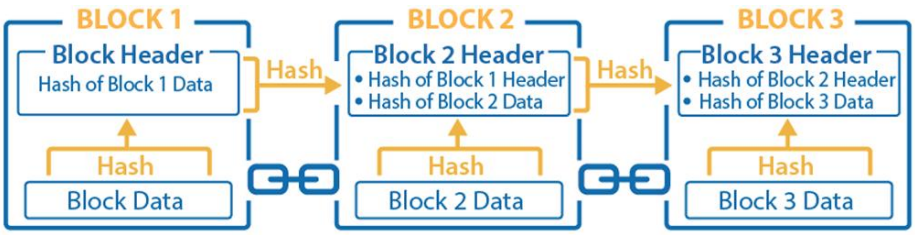
A Blockchain egy elosztott adatbázis-technológia, amely lehetővé teszi a digitális tranzakciók megvalósítását, nyomon követését és rögzítését egy biztonságos és átlátható módon. Az adatbázis egy hálózatba kapcsolt számítógépek több példányában van eltárolva, és az adatokat blokkokba rendezve tartalmazza, amelyek egymásra épülnek, és az adatok érvényességét az előző blokkhoz való kapcsolódással ellenőrzik. Az adatbázis bármely pontjáról elérhető és változtatható, de a blokkok nem változhatnak utólagosan, így biztosítva az adatok integritását és biztonságát. A Blockchain technológia alkalmazása széleskörű lehet, például a pénzügyi tranzakciók, okmányok ellenőrzése és kezelése, egészségügyi információk biztonságos kezelése, az élelmiszerlánc ellenőrzése, a szavazások integritásának biztosítása stb. További figyelemreméltó és érdekes információk széles körben megtalálhatóak a neten pl.: https://aws.amazon.com/what-is/blockchain/?aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc





Centralized Ledger

Distributed Ledger



7.7 Felhő (Cloud) technológia

Jellemzők

Rugalmasság: A felhőtechnológia rugalmasságot kínál az erőforrások bővítéséhez és csökkentéséhez, így az ügyfeleknek csak annyit kell fizetniük, amennyit ténylegesen felhasználnak.

Skálázhatóság: A felhőtechnológia lehetővé teszi az alkalmazások és a rendszerek skálázását, ami biztosítja a nagyobb igénybevételt és a terheléselosztást a rendszerek között.

Hatékonyság: A felhőtechnológia segítségével az ügyfelek számos szolgáltatást használhatnak, amelyekkel javíthatják az üzleti hatékonyságot, például a távoli munkavégzést, az automatizált folyamatokat és a jobb hozzáférést a nagy adatmennyiségekhez.

Költés-hatékonyság: A felhőtechnológia általában alacsonyabb költséggel jár, mint a hagyományos infrastruktúra, mivel az ügyfeleknek nem kell saját hardvert és szoftvert vásárolniuk, üzemeltetniük és karbantartaniuk.

Rugalmas elérhetőség: A felhőtechnológia lehetővé teszi az ügyfelek számára az alkalmazások és az adatok bármikor, bárhonnan való elérését, ami kényelmesebb és hatékonyabb munkavégzést tesz lehetővé.

Biztonság: A felhőtechnológia nagyobb biztonságot kínál, mivel az adatokat a szolgáltatók védik és biztosítják a biztonsági mentéseket és az adatvesztés elleni védelmet.

Innováció: A felhőtechnológia folyamatosan fejlődik és fejleszt, így az ügyfeleknek mindig a legújabb technológiákhoz és szolgáltatásokhoz van hozzáférésük.

8 Irányítás

8.1 Projektvezetési módszertan

8.1.1 IT projekt menedzselési módszerek

Az IT projekt menedzsment módszerek az informatikai projektek tervezését, végrehajtását és ellenőrzését segítik elő, hogy azok hatékonyan és eredményesen valósulhassanak meg.

Néhány példa az IT projektmenedzsment módszerekre

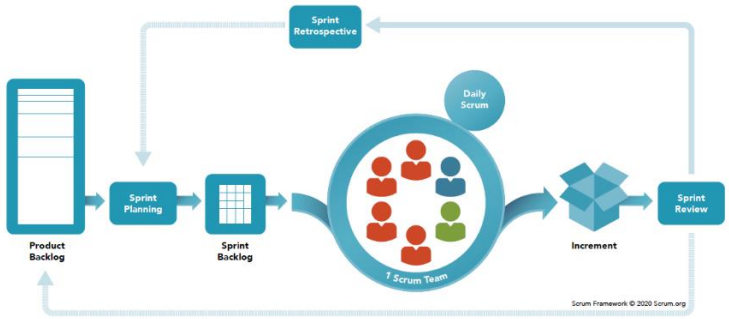
Vízesés modell: Az egyik legelterjedtebb és legrégebbi IT projektmenedzsment módszer, amelynek lépései a következők: igényfelmérés, tervezés, végrehajtás, tesztelés és karbantartás.

Scrum: Egy rugalmas és iteratív fejlesztési folyamat, amely az agilis projektmenedzsment keretein belül működik. A Scrum keretrendszerben az egész projekt egy sor rövid fejlesztési ciklusra van felosztva, amelyeket sprinteknek neveznek.

Kanban: Az agilis projektmenedzsment egy másik módja, amelynek lényege a folyamatábrák és az előrehaladási táblák használata a projektben. A Kanban rendszer lehetővé teszi a csapatok számára, hogy azonnal reagáljanak az előforduló problémákra.

Prince2: A Projects in Controlled Environments (PRINCE2) egy strukturált projektmenedzsment módszer, amely öt fő fázisra osztható: az indítás, az irányítás, az előrehaladás ellenőrzése, a szakasz lezárása és a projekt lezárása.

PMBOK: A Projektmenedzsment Testület útmutatója (Project Management Body of Knowledge) egy projektmenedzsment keretrendszer, amely az öt fázisra van felosztva: a kezdeményezés, a tervezés, a végrehajtás, a felügyelet és az értékelés.



8.1.2 Agilis-alapú projektvezetési módszertanok

Az agilis-alapú projektvezetési módszertanok olyan rugalmas és iteratív megközelítést jelentenek, amelyek célja, hogy a fejlesztési folyamatot felgyorsítsák, az ügyfél igényeihez jobban alkalmazkodjanak, és a fejlesztői csapatok hatékonyságát növeljék. A módszertanok az Agile Manifestóra épülnek, amelynek négy értéke a következő:

- **Együttműködés az ügyféllel a szállítandó érték meghatározásához.**
- **Működő szoftver az elsődleges mérőszám.**
- **Rugalmas változások befogadása az ügyfél igényeinek megfelelően.**
- **Az egyének és az interakciók fontosabbak, mint az eszközök és folyamatok.**

Az agilis-alapú projektvezetési módszertanokat az életszakaszokra bontják, és minden életszakaszban a tervezés, az implementáció, az ellenőrzés és az értékelés során iteratív folyamatokat alkalmaznak. Az agilis módszertanok közé tartozik például a Scrum, a Kanban és az Extreme Programming (XP).

8.1.3 Időgazdálkodás

Az időgazdálkodás olyan tevékenység, amelynek célja, hogy hatékonyan és hatékonyan használjuk az időt, azaz a rendelkezésre álló időt a legjobb módon. Az időgazdálkodás magában foglalja az idő tervezését, szervezését, végrehajtását és ellenőrzését annak érdekében, hogy az időt hatékonyan felhasználjuk a prioritások és a célok eléréséhez.

Az időgazdálkodás fontos szerepet játszik az üzleti életben, mivel az idő az egyik legértékesebb erőforrásunk. Az időgazdálkodás lehetővé teszi, hogy hatékonyabban dolgozzunk, csökkentsük a stresszt és a kiégést, javítsuk a termelékenységet és a hatékonyságot, és elkerüljük az időhátrányt.

Az időgazdálkodás magában foglalja az idő tervezését és szervezését, prioritások beállítását, határidők meghatározását, az idővel való gazdálkodás szokásainak javítását, az időhöz való hozzáállás megváltoztatását, az idő pazarlásának minimalizálását, valamint az időzített szünetek és pihenők beiktatását a hatékonyabb munka érdekében. Az időgazdálkodásnak sokféle technikája van, amelyek közé tartozik például a to-do lista vezetése, a hatékony napirend kialakítása, a prioritások beállítása és az időnyomon követése.

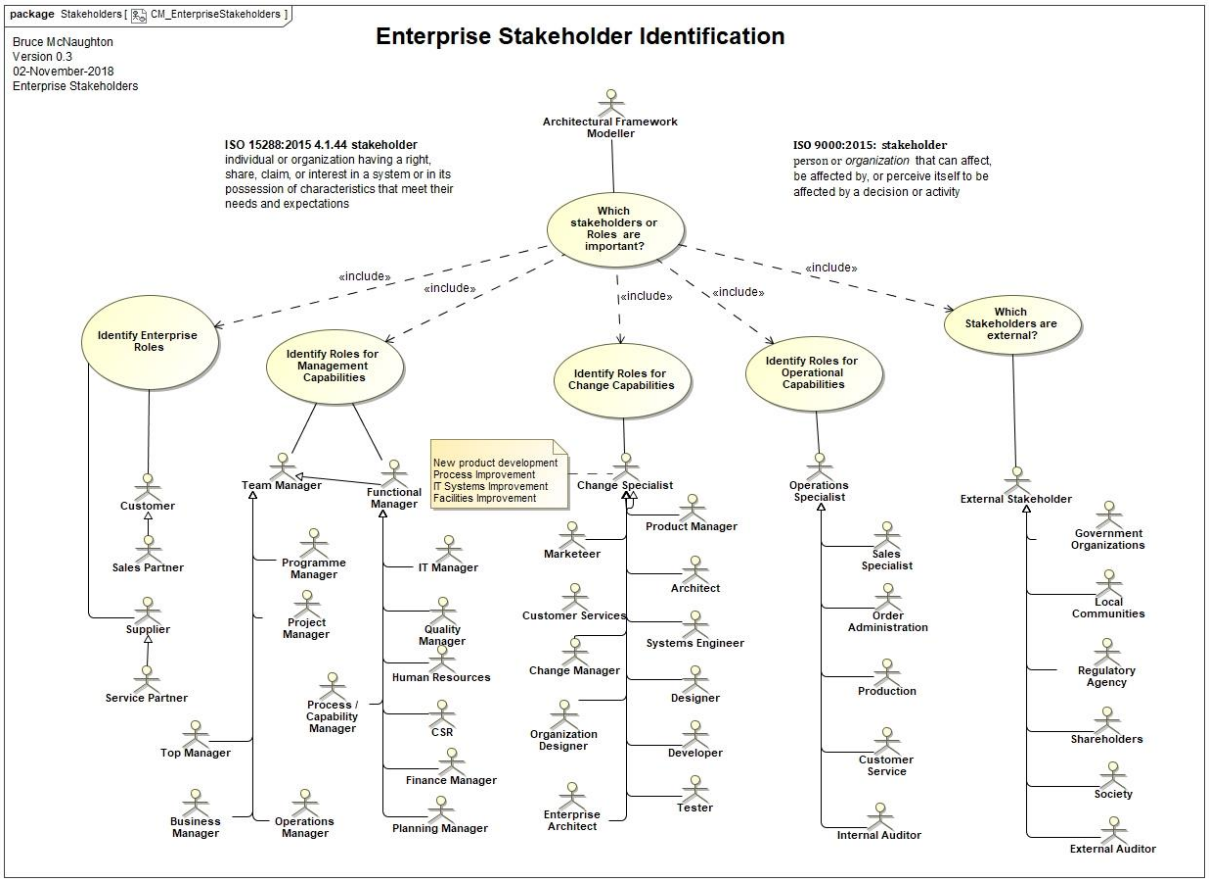
8.2 Kommunikációs elvek

8.2.1 Érdekeltek kezelése

Az "érdekeltek kezelése" (Stakeholder Management) olyan tevékenység, amelynek célja, hogy az érintett feleknek, vagyis az érdekelteknek megfelelő információkat nyújtsunk és kapcsolatot építsünk velük. Az érintettek lehetnek belső (például vezetők, dolgozók) vagy külső (például ügyfelek, beszállítók, szabályozó hatóságok) személyek vagy szervezetek.

Az érdekeltek kezelése fontos része az üzleti tervezésnek és a projektmenedzsmentnek, mivel az érdekelteknek fontos szerepük lehet a projekt sikerében vagy kudarcában. Az érdekeltek kezelése magában foglalja az érintettek azonosítását, az érintettekkel való kommunikációt, az érintettek érdeklődési körének megértését, az érintettekkel kapcsolatos problémák kezelését és az érintettek bevonását a projektbe.

Az érdekeltek kezelése hozzájárulhat a projekt hatékonyságához, a kapcsolatok javításához, a költségek csökkentéséhez, a kockázatok minimalizálásához és a projekt sikeréhez.



8.2.2 Üzleti kommunikáció jellemzői

Az üzleti kommunikáció az üzleti környezetben történő kommunikációra vonatkozik, amely magában foglalja a vállalatok és más szervezetek közötti kommunikációt, valamint az egyének közötti kommunikációt az üzleti környezetben.

Üzleti kommunikáció jellemzők

Célorientáltság: Az üzleti kommunikáció célja a célok elérése, legyen az értékesítés, tárgyalás, tájékoztatás vagy bármilyen más üzleti tevékenység.

Professzionális stílus: Az üzleti kommunikáció szóhasználata és stílusa formális, és magában foglalja a szakmai nyelvezetet, az üzleti fogalmakat és a megfelelő szintű udvariasságot.

Pontosság: Az üzleti kommunikációban nagy hangsúlyt helyeznek a pontos, egyértelmű és világos információk átadására, hogy elkerüljék a félreértéseket és a téves értelmezéseket.

Rugalmasság: Az üzleti kommunikációban fontos a rugalmasság, hogy a kommunikáció hatékony legyen különböző helyzetekben és különböző emberekkel.

Érthetőség: Az üzleti kommunikációban fontos az érthetőség, hogy az üzenetet a címzett könnyen megértse.

Tájékoztatás: Az üzleti kommunikációban fontos az információ megosztása, hogy az üzleti tevékenység hatékonyabb legyen.

Azonnaliság: Az üzleti kommunikációban fontos az időbeni reagálás, hogy az üzleti folyamatok gyorsabbak legyenek.

Együttműködés: Az üzleti kommunikációban fontos az együttműködés, hogy a kommunikáció mindkét fél számára előnyös legyen.

Ezek a jellemzők fontosak az üzleti kommunikáció hatékonysága és sikere szempontjából, és az üzleti környezetben az emberek gyakran tanulják és fejlesztik ezeket a készségeket, hogy hatékonyan kommunikáljanak üzleti partnereikkel és kollégáikkal.

9 Fogalom

9.1 Módszertan

9.1.1 Módszertanok

Az IT módszertanok olyan átfogó tervezési és fejlesztési folyamatok, amelyek segítik a szoftverfejlesztőket és az IT szakembereket a hatékonyabb munkavégzésben és a jobb minőségű alkalmazások és rendszerek létrehozásában. Az IT módszertanok általában alaposan kidolgozott folyamatokat, eljárásokat, irányelveket, eszközöket és sablonokat tartalmaznak, amelyeket a szoftvertervezés és fejlesztés során alkalmazhatunk.

Vízesés modell: A vízesés modell az egyik legelterjedtebb szoftverfejlesztési módszer, amely egy lineáris, lépésenkénti megközelítést követ. Az egyes fázisok egymást követik és csak akkor kezdődnek el, amikor az előző fázis befejeződött. Az előnye, hogy szigorúan szervezett és dokumentált, hátránya pedig, hogy a fejlesztési folyamat rugalmatlan és nehezen illeszthető az ügyfél igényeihez.

Agile módszertan: Az Agile módszertanok olyan iteratív és inkrementális megközelítést alkalmaznak, amelyek lehetővé teszik a fejlesztőknek, hogy gyorsabban és hatékonyabban alkalmazzák az ügyfél visszajelzéseit. Az Agile módszertanok lehetővé teszik az ügyfél igényeinek gyorsabb és rugalmasabb beépítését az alkalmazás vagy rendszer fejlesztési folyamatába.

DevOps: A DevOps egy olyan módszertan, amely lehetővé teszi a szoftvertervezőknek és a szoftverüzemeltetőknek, hogy együttműködjenek és az alkalmazásokat a lehető leghatékonyabb módon működtessék. A DevOps az automatizált tesztelést és az állandó integrációt használja a szoftvertervezési és szoftverüzemeltetési folyamatok hatékonyabbá tételéhez.

Lean: A Lean módszertan az erőforrások optimalizálására és a felesleges veszteségek csökkentésére összpontosít. A Lean módszertan az állandó javításra és a folyamatok folyamatos fejlesztésére összpontosít.

Ezek az IT módszertanok segítenek a szoftvertervezőknek és a szoftverfejlesztőknek hatékonyabbá tenni a fejlesztési folyamatot.

A módszertanok időrendi sorrendje

1. Strukturált programozás
2. "Vízesés"
3. Iteratív, Inkrementális
4. Spirális
5. V-Modell
6. RAD
7. Unified Process
8. Scrum
9. Agile
10. PSE

9.2 Architektúra

9.2.1 Informatikai Architektúra

Az informatikai architektúra egy olyan fogalom, amely a számítógépes rendszerek és alkalmazások tervezésének és felépítésének strukturális és szervezeti keretrendszerét írja le.

Az informatikai architektúra fogalma átfogóan foglalkozik a rendszerek összetevőivel, azok viszonyaival, illetve a rendszerek működésének és fejlesztésének módszertanával.

Az informatikai architektúra általános területei

1. **Rendszerarchitektúra**: Ez a szint az informatikai rendszerek alapvető struktúrájával foglalkozik. Ide tartozik a hardverkomponensek, szerverek, hálózati infrastruktúra és adatbázisok elrendezése és kapcsolódása. A rendszerarchitektúra meghatározza a rendszer működési módját, a komponensek közötti kommunikációt és az adatfolyamokat.

2. **Alkalmazásarchitektúra**: Ez a szint az alkalmazások, szoftverek és szolgáltatások tervezésével és elrendezésével foglalkozik. Az alkalmazásarchitektúra meghatározza az alkalmazások komponenseit, azok működési logikáját, az adatbázis-kezelést, az interfészeket és a felhasználói felületeket.

3. **Adatarchitektúra**: Ez a szint az adatok tárolásával, kezelésével és az adatbázisokkal kapcsolatos tervezési döntésekkel foglalkozik. Az adatarchitektúra meghatározza az adatmodelleket, az adatbázis-struktúrát, adatazonosítókat és az adatintegritást.

4. **Technológiai architektúra**: Ez a szint a technológiai eszközökkel és platformokkal foglalkozik, amelyek támogatják az informatikai rendszerek működését. Ide tartoznak a hardverek, operációs rendszerek, adatbázis-kezelő rendszerek, hálózati infrastruktúra és egyéb technológiai komponensek.

Az informatikai architektúra célja az, hogy meghatározza a rendszerek struktúráját és viszonyait annak érdekében, hogy a rendszerek hatékonyan működjenek, könnyen karbantarthatóak legyenek, skálázhatóak legyenek és a felhasználói igényeknek megfelelően fejleszthetők legyenek. Az architektúra tervezése sor

9.2.2 Architektúra Típusok

Üzleti architektúra

A jól megfogalmazott üzleti architektúra az átfogó vállalati architektúra sikeres eredményének sarokköve. Meghatározza az üzleti mozgatórugókat, az üzleti stratégiát, a működési modelleket, a célokat és célkitűzéseket, amelyeket a szervezetnek el kell érnie ahhoz, hogy át tudjon lépni egy potenciálisan versenyképes és zavaró üzleti környezetbe. A többi építészeti szakterületen dolgozó építészeknek meg kell érteniük az üzleti architektúrát, amely a saját architektúraleírásuk alapja, és útmutatóul szolgál az elérendő üzleti eredményekhez.

Az üzleti architektúra jellemzően az alap- és a célarchitektúrák leírásából, valamint a végrehajtható átmenetek meghatározásából áll, amelyeket az ütemtervi diagramokon írnának le.

Információs architektúra

Az információs architektúra kulcsfontosságú a vállalati architektúra program sikeréhez, mivel az információt a többi architektúrát alkotó komponensek hozzák létre, fogyasztják és semmisítik meg. Annak megértése, hogy mely üzleti funkciók és folyamatok használnak információt, mely alkalmazások szolgálnak törzsadatként, hol jön létre és semmisül meg az információ, és mely technológiai összetevők tárolják és kezelik az információt, kritikus fontosságú az üzleti eredmények eléréséhez.

Alkalmazási architektúra

Az alkalmazásarchitektúra fontos katalógus a vállalaton belüli alkalmazásokról, leírva az információk átalakítása, továbbítása és tárolása érdekében végzett munkájukat. Az architektúra leírja továbbá az alkalmazások által megkövetelt vagy biztosított interfészeket, valamint azt, hogy az alkalmazások hogyan lépnek kölcsönhatásba egymással az üzleti modellekben - például az üzleti folyamatábrákban és a képességmodellekben - leírt tevékenységek elvégzése érdekében. Az alkalmazások katalógusát, az interfészeket és az ezek kölcsönhatását leíró diagramokat és mátrixokat csak egyszer kell meghatározni a vállalati szinten. Az alkalmazás-architekt képes lesz a meglévő artefaktumok e leltárára támaszkodni új architektúrák létrehozásához, az alap- és potenciálisan a jövőbeli állapot-architektúra részeként besorolva azokat. Ha egy architektúra új alkalmazásokat vezet be, ezeket hozzá lehet adni a célállapot leírásához.

Technológiai architektúra

A technológiai architektúra a többi architektúra alapját képezi, leírást adva a logikai, fizikai és virtuális infrastruktúráról, amely támogatja az alkalmazási szolgáltatások végrehajtását, amelyek viszont az információs és üzleti funkciókat és szolgáltatásokat támogatják.

Biztonsági architektúra

A biztonsági architektúra biztonsági szempontból az összes többi architektúra egy szeletét jelenti. Azért szerepel különálló architektúraként, mert fontos szerepet játszik abban, hogy a vállalati biztonsági irányelvek az architektúrán keresztül megvalósuljanak. A biztonság megsértése az üzleti architektúrától a technológiai architektúrán keresztül bármelyik ponton bekövetkezhet. Ez magában foglalhatja annak bemutatását, hogy az architektúrák hogyan felelnek meg a vállalat által közzétett vagy az iparági megfelelési szabályok részeként rendelkezésre álló biztonsági ellenőrzéseknek.

9.2.3 Architektúra Mintázat vs. Architektúra Típus

Az Architektúra Minták és az Architektúra Típusok eltérő módon osztják fel az alkalmazások és rendszerek architektúráját.

Architektúra Típusok (Architecture Type)

Általában architekturális elrendezéseket határoznak meg, amelyek az alkalmazások és rendszerek tervezésének kiindulópontjaként szolgálnak. Az architektúra típusok olyan alapvető architekturális alapokat írnak elő, mint például a szolgáltatás-orientált architektúra (SOA), a többrétegű architektúra vagy a mikroszolgáltatások.

Az Architektúra Típusok olyan magas szintű tervezési elvek és irányelvek, amelyek megfelelőek lehetnek az alkalmazások és rendszerek általános céljainak megvalósításához.

Architektúra Minták (Architecture Pattern)

Konkrétabb architekturális megoldásokat kínálnak az alkalmazások és rendszerek tervezési problémáira. Az Architektúra Minták tipikus tervezési sablonok, amelyeket az alkalmazások és rendszerek tervezése során ismételten alkalmazhatunk.

Az Architektúra Minták konkrét, megszokott megoldásokat kínálnak olyan tervezési problémákra, mint az adatbázis-kezelés, az architektúra kialakítása, a rendszer biztonságossá tétele, a teljesítmény növelése stb.

Tehát az Architektúra Típusok általánosabb, magas szintű tervezési elveket írnak elő, míg az Architektúra Minták specifikusabb, konkrétabb megoldásokat nyújtanak. **Mindkettő fontos szerepet játszik az alkalmazások és rendszerek hatékony tervezésében és fejlesztésében.**

9.2.4 Architektúra Keretrendszerek

Az Architektúra Keretrendszer (Architecture Framework) egy összetett struktúra, amely :

- Az üzlet, az információ, az emberek, a folyamatok és a technológia közötti összetett kölcsönhatásokat jeleníti meg
- A keretrendszer elemeinek kölcsönösen kizárónak és együttesen teljesnek kell lenniük
- A (vállalati architektúra) keretrendszerek és a stratégiai tervezés a kulcsa a szervezet üzleti célkitűzései tervezésének, koordinálásának és megvalósításának

9.2.4.1 Szerepe

Strukturált megközelítést nyújt az informatikai rendszerek tervezéséhez és dokumentálásához, és lehetővé teszi az informatikai rendszerek és alkalmazások átlátható és összehasonlítható módon történő tervezését.

Az Architecture Framework lehetővé teszi az informatikai rendszerek és alkalmazások egységes és standardizált megtervezését, amely hozzájárul a jobb minőségű informatikai rendszerek létrehozásához, a fejlesztési folyamatok hatékonyságának növeléséhez, és a költségek csökkentéséhez.

9.2.4.2 A keretrendszer elemei

Részletes rendszer architektúra

Meghatározza a részletes rendszer architektúrát, beleértve az alkalmazások, szolgáltatások és rendszerek részletes tervezését.

Implementációs rendszer architektúra

Meghatározza a rendszer implementációs architektúráját és az implementációhoz szükséges eszközöket és folyamatokat.

Alapvető modell

A rendszer alapvető modelljét és elemeit határozza meg.

Általános rendszer architektúra

Meghatározza a rendszer általános architektúráját és tervezési alapelveit.

Szabványosítás - TOGAF

Architektúra keretrendszer (Architecture Framework) a legelfogadottabb, "de facto" szabványa a TOGAF - The Open Group Architecture Framework.

9.2.5 Architektúra kontextusa

A vállalati architektúra egy program szintű tevékenység, és emiatt leginkább úgy lehet tekinteni, mint az üzleti tevékenység egy operatív egységére, és mint ilyen, van kontextusa. A programnak értéket kell nyújtania az üzlet számára; ezt úgy éri el, hogy biztosítja, hogy az architektúrával kapcsolatos erőfeszítések összhangban legyenek a szervezet stratégiai terveivel, és hogy a megvalósítási kezdeményezések úgy valósuljanak meg, hogy tiszteletben tartsák a vállalati architektúrát.

Stratégiai kontextus

Az üzleti architektúrát vissza kell kapcsolni a stratégiai tervekhez annak biztosítása érdekében, hogy az architektúrát részletesebben leíró összes többi architektúra-tartomány végső soron a vállalat javát szolgálja és értéket teremtsen. Az üzleti architektek jellemzően a vállalati stratégáktól gyűjtik az információkat, és beavatottaknak kell lenniük a vállalat és szervezeteinek jövőbeli terveiről szóló magas szintű megbeszélésekbe és döntésekbe.

Végrehajtási kontextus

Az architektúra célja annak biztosítása, hogy a kezdeményezések és projektek az architektúra modellekben leírt üzleti értéket és előnyöket biztosítsák, ezért a megvalósítási projektek megfelelőségének ellenőrzése kritikus fontosságú az architektúrák és végső soron az architektúra program sikere szempontjából.

A megvalósítás irányítása az architektúra folyamatának kulcsfontosságú része, amelyet hivatalosan is kezelni kell annak biztosítása érdekében, hogy az architektúra útmutatóul szolgáljon a megvalósítási csapatok számára, de annak biztosítása érdekében is, hogy az architektúrát egyértelműen megértsék és kövessék.

Végső soron a megvalósítási kezdeményezések alakítják át a szervezetet az alapállapotból (jelenlegi állapotból) a célállapotba (jövőbeli állapotba), és a program sikere szempontjából kritikus fontosságú annak biztosítása, hogy ezek a kezdeményezések megfeleljenek az alapelveknek és a terveknek.

9.3 Implementáció

9.3.1 „Ripple Effect” (hullámeffektus)

A szoftvertervezésben használt fogalom, amely azon változásokra utal, amelyeket egy rendszer egy részében végrehajtanak, és amelyek hatással lehetnek a rendszer más részeire is. A változások terjedése olyan, mint a hullámok a vízben, és általában nem előre jelezhetők.

A Ripple Effect nagyon fontos a szoftvertervezésben, mivel egy változtatás egy kódrészletben hatással lehet a teljes rendszerre. Ez azt jelenti, hogy egy kis változás is nagyobb hatást gyakorolhat a rendszerre, és akár hibákat is okozhat.

A Ripple Effect kezelése érdekében a szoftvertervezőknek figyelembe kell venniük a rendszer egészének összefüggéseit, és előre kell tervezniük a változások hatásait. Ezt általában tesznek a moduláris tervezéssel, a tervezési mintázatok alkalmazásával és az egységtesztelési módszerekkel.

A Ripple Effect kezelése kulcsfontosságú az Agilis fejlesztési módszerekben is, mivel ezekben a módszerekben a változtatások gyakran történnek, és gyorsan kell reagálni a változó üzleti igényekre.

9.4 Technológia

9.4.1 Az IT üzemeltetés eszközei

Monitorozó eszközök: ezek az eszközök nyomon követik az IT rendszerek teljesítményét, figyelik a rendszer állapotát, és jelzik az esetleges problémákat. Ide tartoznak az általános rendszerfigyelők, hálózati monitorok, adatbázis-monitorok, alkalmazásfigyelők stb.

Automatizálási eszközök: olyan eszközök, amelyek automatizálják az IT rendszerekkel kapcsolatos feladatokat, például a frissítéseket, az alkalmazások telepítését vagy a biztonsági mentéseket. Ide tartoznak a konfigurációkezelők, az automatizált telepítési eszközök, a scipt nyelvek és a feladatütemezők.

Tároló- és backup eszközök: az IT rendszereknek tárolóeszközökre van szükségük az adatok tárolásához és biztonsági mentésekhez. Ide tartoznak az adattároló rendszerek, a felhőtárolás, az archiváló rendszerek és a backup eszközök.

Virtualizációs eszközök: az IT infrastruktúra virtualizálása lehetővé teszi, hogy az üzemeltetők több virtuális környezetet hozzanak létre egyetlen fizikai szerveren vagy tárolórendszeren. Ide tartoznak a virtualizációs platformok, az alkalmazás virtualizációs eszközök és a desktop virtualizáció.

Incidens management eszközök: olyan eszközök, amelyek lehetővé teszik az IT üzemeltetők számára, hogy hatékonyan kezeljék az incidenseket és problémákat. Ide tartoznak az incidenskezelő rendszerek, a hibajelentő eszközök és a problémakezelő eszközök.

Konfigurációkezelő eszközök: az IT rendszerek állandó konfigurációs változásoknak vannak kitéve, amelyek kezelése nagyon nehéz lehet. Az ilyen eszközök lehetővé teszik az IT üzemeltetők számára, hogy könnyen és hatékonyan kezeljék a konfigurációs változásokat és visszaállítsák a korábbi konfigurációt. Ide tartoznak a konfigurációkezelő eszközök és a változatkezelő rendszerek.

9.4.2 Database Reengineering

A Database Reengineering az adatbázis-tervezési folyamatnak az a funkciója, amely lehetővé teszi a meglévő adatbázisok feljavítását, átalakítását és modernizálását, javítva a teljesítményt, a megbízhatóságot és a skálázhatóságot.

Az adatbázisokat idővel általában új igényeknek kell megfelelően fejleszteni és módosítani kell őket. A Database Reengineering funkció lehetővé teszi a meglévő adatbázisok felülvizsgálatát és újra tervezését annak érdekében, hogy megfeleljenek az újabb igényeknek és elvárásoknak. Ez magában foglalhatja a már meglévő adatok struktúrájának átalakítását, a táblák újrafelosztását, az indexek és a kulcsok módosítását, valamint az adatbázis optimalizálását az újabb adatok kezelése érdekében.

A Database Reengineering funkció segít abban, hogy az adatbázisok a lehető legjobban működjenek, és lehetővé teszi az adatok megbízhatóbb és hatékonyabb kezelését. Ennek eredményeként az üzleti folyamatok hatékonyabbak és megbízhatóbbak lesznek, amelyek javítják az üzleti teljesítményt és növelik a vállalkozás hatékonyságát.

9.4.3 Database Reverse Engineering

A Database Reverse Engineering funkció az adatbázis-tervezési folyamatnak az a része, amely lehetővé teszi a meglévő adatbázisok struktúrájának, szabályainak és relációinak visszafejtését és dokumentálását. Ez azért fontos, mert a dokumentáció nélküli, összetett adatbázisok nehezen kezelhetők, nehéz hibakeresési folyamatokat eredményeznek, és akadályozhatják a további fejlesztéseket.

A Database Reverse Engineering segítségével az adatbázisok struktúrája és relációi tisztázhatóak és átláthatóvá válnak. Ennek eredményeként a fejlesztők képesek lesznek könnyebben és hatékonyabban kezelni az adatbázist, javítani a hibákat, és az új funkciókat gyorsabban implementálni.

Ezen kívül, a Database Reverse Engineering funkció lehetővé teszi az adatbázis tervezőinek és fejlesztőinek, hogy pontos dokumentációt készítsenek az adatbázis struktúrájáról, ami nagyban segíti az adatbázis karbantartását és a további fejlesztéseket. Ez segíthet az üzleti folyamatok javításában és a hatékonyabb döntéshozatalban.

9.5 Biztonság

9.5.1 GDPR

A GDPR az Európai Unió által 2018. május 25-én bevezetett általános adatvédelmi rendelet, amely a személyes adatok védelmére vonatkozik az Európai Gazdasági Térségen belül. Az GDPR célja az egyének személyes adatainak védelme, valamint a vállalkozások és az állami szervezetek által kezelt adatok biztonságának biztosítása.

Az GDPR a személyes adatok kezelésével kapcsolatos jogokat és kötelezettségeket határozza meg, ideértve az adatvédelmi nyilatkozatokat, a beleegyezést, a tájékoztatást és az adatvédelmi jogokat. Az adatkezelőknek kötelezően be kell tartaniuk az adatvédelemre vonatkozó szabályokat, valamint biztosítaniuk kell, hogy az egyének könnyen hozzáférhessenek az adataikhoz, és azokat meg is lehessen változtatni.

Az GDPR-nak számos előnye van, ideértve a jobb adatvédelmet és adatbiztonságot, valamint az egyének számára nagyobb biztonságot és ellenőrzést ad az adataik felett. Azonban az GDPR kötelező betartása jelentős kihívásokat jelenthet az üzleti szervezetek számára, például az adatvédelmi rendszerek kialakítása, az adatok nyomon követése és az adatvédelmi jogszabályok betartása.

9.5.1.1 Mit nem határoz meg a GDPR ?

Bár az általános adatvédelmi rendelet (GDPR) sokféle követelményt tartalmaz az adatvédelem és adatkezelés területén, bizonyos területeken azonban nincs hatásköre. Ezek közé tartoznak:

Az adatkezelés általános szabályai: Az GDPR nem határozza meg az adatkezelés általános szabályait, vagyis azt, hogy mikor és hogyan lehet adatokat kezelni. Az ilyen szabályokat az egyes országoknak kell meghatározniuk a nemzeti jogszabályaikban.

A nem személyes adatok kezelése: Az GDPR csak a személyes adatok kezelésére vonatkozik, nem pedig a nem személyes adatokra.

Állami szervek: Az állami szervek által végzett adatkezelés tekintetében az Európai Unió egyes tagállamai továbbra is rendelkezhetnek külön nemzeti jogszabályokkal.

Nem uniós országok: Az GDPR csak az Európai Unióban és az Európai Gazdasági Térség országaiban érvényes, és nem vonatkozik a nem uniós országokra.

Kriminális ügyek: Az adatkezelésre vonatkozó jogszabályokat külön szabályozzák a kriminális ügyekben, és ezek a szabályok az GDPR hatálya alól kivételek.

Összességében az GDPR csak a személyes adatok kezelésével kapcsolatos jogokra és kötelezettségekre vonatkozik, és nem határozza meg az adatkezelés általános szabályait. A nem személyes adatokra, az állami szervek által végzett adatkezelésre és a kriminális ügyekre külön jogszabályok vonatkoznak.

9.5.2 Anonimizálás

Az anonimizálás az adatvédelem egyik módszere, amely során az adatokból azonosító adatokat (például név, születési dátum, cím stb.) eltávolítják vagy kódolják, így azok nem lehetnek összekapcsolhatók egy adott személlyel. Az anonimizálás azt jelenti, hogy az adatokat olyan formában dolgozzák fel, hogy azok ne lehessenek azonosítani, így azokat az adatkezelők szabadon felhasználhatják kutatási vagy elemző célra.

Az anonimizálás két fő típusa van: az egyszerű és a magas szintű anonimizálás. Az egyszerű anonimizálás az azonosító adatok eltávolítását jelenti, például a nevet, a címet vagy a születési dátumot. A magas szintű anonimizálás az adatok további szűrését jelenti, például az adatok részleges elmosását vagy kódolását.

Az anonimizálás fontos szerepet játszik az adatvédelemben, mivel azok az adatok, amelyek nem tartalmaznak azonosító adatokat, nem tekinthetők személyes adatoknak. Azonban fontos megjegyezni, hogy az anonimizálás sem garantálja teljes mértékben az adatok védelmét, mivel néhány adat még mindig összekapcsolható lehet más adatforrásokkal. Ezért az adatvédelmi jogszabályok további biztonsági intézkedéseket írnak elő az adatok védelme érdekében.

9.5.3 Számítógéprendszerek behatolásvédelme

A Számítógéprendszerek behatolásvédelme az információbiztonság egyik fontos eleme, amely célja, hogy megakadályozza a számítógépes rendszerekbe történő jogosulatlan belépést és az azokon történő jogosulatlan tevékenységeket.

Az adatvédelem és az adatbiztonság fontosak a számítógépek és a hálózatok számára, és az adatvédelmi rendszerek megakadályozzák az illetéktelen felhasználókat a számítógépes rendszerekhez való hozzáférésben, az adatokhoz való hozzáférésben és az adatok manipulálásában.

Számítógéprendszerek behatolásvédelmének stratégiája

Az első rész a megelőzés, amely célja, hogy megakadályozza a behatolást, és az illetéktelen hozzáférési kísérletek észlelése és blokkolása. A megelőzési lépések magukban foglalhatják a tűzfalakat, a vírusvédelmet, az erőteljes jelszavak használatát, az illetéktelen hozzáférés megakadályozását, a hozzáférési jogosultságok korlátozását és a rendszeres biztonsági frissítéseket.

A második rész a behatolás észlelése és az incidenskezelés. Ez a rész célja az illetéktelen belépések és tevékenységek észlelése és az azokra való reagálás. Ez magában foglalja az eseménynaplók használatát, az események elemzését és értékelését, valamint a helyreállítási tervek kidolgozását, amelyek lehetővé teszik az üzleti folyamatok gyors és hatékony helyreállítását.

Összességében a Számítógéprendszerek behatolásvédelme kulcsfontosságú a számítógépes rendszerek és a hálózatok biztonságának megőrzéséhez, és a megfelelő megelőzési és észlelési intézkedések lehetővé teszik a károk minimalizálását és a folyamatok zavartalan működését.

9.6 Menedzselés

9.6.1 IT menedzselés

Az IT menedzsment az információs technológia átfogó irányítását jelenti, beleértve az erőforrások kezelését, a projektek tervezését és végrehajtását, a folyamatok optimalizálását és a kockázatkezelést. Az IT menedzsment célja, hogy biztosítsa az IT szolgáltatások hatékony és hatékony működését, valamint az üzleti célok támogatását azáltal, hogy biztosítja a szükséges eszközöket, erőforrásokat és eljárásokat az IT tevékenységek kezeléséhez. Az IT menedzsment magában foglalja az IT stratégia kidolgozását, a projektmenedzsmentet, az infrastruktúra menedzsmentet, a szolgáltatás menedzsmentet és az információbiztonságot.

9.6.2 IT stratégia

Az IT stratégia elemei

Üzleti igények: Az IT stratégiát az üzleti igények határozzák meg, amelyeket a vállalatnak kielégítenie kell.

Rendszerintegráció: Az IT stratégia megtervezésekor figyelembe kell venni a vállalat jelenlegi rendszereit és azok összekapcsolhatóságát.

Technológiai irányelvek: Az IT stratégia meghatározza, milyen technológiai irányelvek mentén kell dolgozni a vállalatnak.

Szolgáltatások: Az IT stratégia meghatározza a vállalat által nyújtott szolgáltatásokat és a szolgáltatások minőségét.

Biztonság: Az IT stratégia magában foglalja az adatbiztonsággal és a kiberbiztonsággal kapcsolatos intézkedéseket.

Infrastruktúra: Az IT stratégia meghatározza az infrastruktúra elemeket, mint például a hardverek, szoftverek és adatközpontok.

Rendszerfelügyelet: Az IT stratégia meghatározza a rendszerfelügyeleti folyamatokat, amelyek biztosítják az üzemeltetési folyamatok megfelelő működését.

Költségek: Az IT stratégia meghatározza, milyen költségekkel kell számolni az IT területén, és hogyan lehet a kiadásokat minimalizálni.

Stratégiai partnerségek: Az IT stratégia meghatározza a stratégiai partnerségeket, amelyekre a vállalat támaszkodhat a technológiai fejlesztésekben.

Ezek az elemek változó mértékben szerepelhetnek az IT stratégiákban, és az egyes vállalatoknak a saját igényeikhez kell igazítaniuk azokat.

9.6.3 IT architektúra menedzsment

Az IT-architektúra menedzsment az információtechnológiai (IT) architektúrák tervezésének, irányításának és fenntartásának folyamata a vállalaton belül. Ennek a folyamatnak a célja, hogy biztosítsa az IT rendszerek hatékony működését, összhangban az üzleti célokkal és az IT-stratégiával. Az IT-architektúra menedzsment feladatai közé tartozik az IT-architektúra stratégiai tervezése, az IT-architektúra szabványok kidolgozása és az IT-architektúra értékelése és ellenőrzése. Emellett az IT-architektúra menedzsment felelős az IT-infrastruktúra tervezéséért és a szervezet belső szabályainak megfelelő alkalmazásáért.

9.6.4 IT infrastruktúra menedzsment

Az IT infrastruktúra menedzsment a hardver, szoftver és hálózati erőforrások hatékony és hatékony kezelése a vállalati szervezetben. Az IT infrastruktúra menedzsment magában foglalja a szerverek, a hálózatok, a tárolórendszerek, a felhőszolgáltatások, a szoftverek és az adatbázisok kezelését, telepítését, karbantartását és frissítését is. A megfelelő IT infrastruktúra menedzsment lehetővé teszi a szervezet számára, hogy hatékonyan és hatékonyan működjön, javítsa a termelékenységet és csökkentse az IT-beli hibákból adódó károkat.

9.6.5 IT Incidens kezelés

Az incidenskezelés az az ITIL (Information Technology Infrastructure Library) szabványban meghatározott folyamat, amelynek célja a szervezetnél bekövetkezett, nem tervezett események kezelése, hogy a normál működés mielőbb helyreálljon. Az incidenskezelés magában foglalja az események rögzítését, azok értékelését, a probléma diagnosztizálását és megoldását, a dokumentációt és a riportokat. Az incidenskezelés fontos része az IT szervezetnek, mivel biztosítja a szolgáltatások zavartalan működését és csökkenti az ügyfél elégedetlenségét.

9.6.5.1 Incidens kezelő eszközök

Az Incident Management az informatikai rendszerek és alkalmazások üzemeltetésének egyik folyamata, amely az előre nem tervezett események kezelésére fókuszál. Ennek során különböző eszközök segítségével történik az incidensek rögzítése, kezelése és dokumentálása. Néhány példa ilyen eszközökre:

- ServiceNow
- Jira Service Desk
- BMC Remedy
- Zendesk
- Freshservice
- SolarWinds Service Desk
- PagerDuty
- OpsGenie
- VictorOps
- Splunk

9.7 Szabályozás

9.7.1 Governance

A governance a szervezetek, illetve azok tagjai közötti döntéshozatali, felelősségvállalási és irányítási folyamatokat jelenti. Általában a vállalatok, szervezetek belső irányítási rendszereit jelenti, amelyek segítenek biztosítani az átláthatóságot, a szabályozást és a hatékony döntéshozatalt a szervezeti célok elérése érdekében. A governance egy átfogó keretrendszer, amely magában foglalja a stratégiák kidolgozását, a szervezeti irányítást, a kockázatkezelést és a teljesítménymérési folyamatokat. Az IT governance kifejezés specifikusan az IT-szervezetek belső irányítási rendszerét jelenti.

9.7.2 IT Governance

Az IT governance olyan irányítási rendszer, amely a vállalati stratégia és a technológiai infrastruktúra közötti összhangot biztosítja. Az IT governance célja, hogy segítsen a vállalatnak hatékonyan és biztonságosan használni az IT erőforrásait, csökkentve a kockázatokat és biztosítva a vállalati értéket. Az IT governance általában magában foglalja az IT stratégiát, a projektportfólió kezelést, a kockázatkezelést, a pénzügyi tervezést és ellenőrzést, a szabályozást és a megfelelőséget, valamint az IT teljesítményének értékelését és felügyeletét.

9.7.3 IT irányelvek

Az IT irányelvek olyan dokumentumok vagy szabályok, amelyek meghatározzák, hogy az adott szervezet hogyan használja, kezeli és védi az IT erőforrásait. Az IT irányelvek segítenek az IT szervezetnek és a felhasználóknak egyaránt megérteni a szervezet IT szabályait és az IT erőforrásokkal kapcsolatos legjobb gyakorlatokat. Az IT irányelvek az adatvédelem, az IT biztonság, az IT folytonosság, az adatkezelés, az adatmegőrzés, a hozzáférés-kezelés, az IT szolgáltatások és az IT kockázatkezelés terén határozzák meg a szervezet irányvonalait és előírásait. Az IT irányelvek célja, hogy segítsék a szervezetet az IT erőforrások hatékony használatában és a szervezetben lévő IT kockázatok minimalizálásában.

9.7.4 IT rendszerfelügyeleti folyamatok

Az IT rendszerfelügyeleti folyamatok a számítástechnikai rendszerek folyamatos üzemeltetésének biztosításához szükséges tevékenységeket foglalják magukban.

Rendszerfelügyeleti folyamatok

Monitorozás: A rendszerek, alkalmazások, hálózatok folyamatos figyelése, ellenőrzése azért, hogy időben észlelhessük az esetleges hibákat, teljesítményproblémákat vagy biztonsági veszélyeket.

Diagnosztizálás: Azonosítja a problémát, megállapítja annak okát és a megoldáshoz szükséges lépéseket.

Hibaelhárítás: A probléma megoldása és az eredeti állapot helyreállítása, ideiglenes megoldások alkalmazása, amíg az alapvető problémát meg nem oldjuk.

Frissítések és változtatások menedzselése: Az új frissítések, patchek, biztonsági frissítések és változtatások telepítése, bevezetése a rendszerbe.

Backup és restore: Az adatok biztonsági mentése és azok visszaállítása, ha valamilyen katasztrófa vagy adathiba miatt elvesznek.

Személyzet menedzsment: Az IT személyzet felügyelete, vezetése, a megfelelő képzés és továbbképzés biztosítása.

Szolgáltatás szintű megállapodások menedzselése: A szolgáltatás szintjének megfelelő teljesítmény biztosítása, a szolgáltatási szint megállapodások betartása és teljesítése.

Kapacitás tervezés és menedzsment: Az IT infrastruktúra kapacitásának tervezése, értékelése és fenntartása annak érdekében, hogy az üzleti igényeknek megfelelően működjön a rendszer.

Ezek a folyamatok biztosítják az IT rendszerek biztonságos és zavartalan működését.

9.7.5 Szabványok, ajánlások, tanúsítások

Az informatikai technológiai szabványok és szakági ajánlások ismerete rendkívül fontos az IT tervezők számára. A tervezőnek el kell tudni igazodnia a szoftver rendszerekre, a hardver technológiákra, az informatikai biztonságra és adatvédelemre, a folyamatokra, az informatikai szolgáltatásnyújtásokra és minőségbiztosításra vonatkozó nemzetközi és hazai szabványok és ajánlások rengetegében mind az általános érvényű szabványok mind az adott domain/szakterületre vonatkozó előírások tekintetében. Ez a felkészítő anyag terjedelme miatt sem alkalmas mindnek a felsorolására, így itt elsősorban a legszükségesebb szabvány-sorozatok tematikáját és a forrásokat tudjuk megadni, hogy hol találhatóak meg a keresett műszaki ajánlások.

Minden **IT tervezőnek** az előkészítési szakaszban **fel kell mérnie az adott feladathoz szükséges, akkor hatályos jogszabályi és technológiai szabályozási környezetet** és a **tervezési dokumentumban** ezeket **rögzíteni kell.**

9.7.6 Nemzetközi technológiai ajánlások és szabványok

Szoftver-intenzív rendszerek architektúra leírási szabványainak célja a nagy bonyolult rendszerek, a beágyazott rendszerek, a rendszerek rendszerei, valamint az információrendszerek szabványos formátumú leírásához szükséges alapok megteremtése. A legfontosabb nemzetközi IT-t érintő szabványok, sorozatok:

SZABVÁNY / AJÁNLÁS / TANÚSÍTÁS	TARTALOM
IEEE 1471 standard	A "szoftver-intenzív rendszer" architektúra meghatározása
ISO / IEC / IEEE 42010	A rendszer- és szoftverfejlesztés architektúra leírásainak keretrendszere (nemzetközi szabványa) • Az architektúra leírások felhasználásával a rendszerek architektúrájának létrehozásával, elemzésével és fenntartásával foglalkozik • Definiálja az architektúra leírás fogalmi modelljét • Meghatározza az architektúra leírásához szükséges tartalmakat • Meghatározza az architektúra nézőpontok, keretek és leírási nyelvek szükséges tartalmát • A keretrendszer egy architektúra leírás tartalmát szabványosítja, két fő szemszögből történő megközelítést használ fel erre a célra, a nézetet (view) illetve a nézőpontot (viewpoint). A nézet azt jeleníti meg, amit a kiválasztott nézőpontból látni lehet • A mellékletek a fő fogalmakat, a terminológia motivációját és hátterét, valamint alkalmazási példákat mutatnak be
Common Criteria (Common Criteria for Information Technology Security Evaluation = CC)	Az informatikai termékek és rendszerek információs technológia biztonsági értékelése

SZABVÁNY / AJÁNLÁS / TANÚSÍTÁS	TARTALOM
COBIT (Control Objectives for Information and related Technology)	Az informatikai biztonság irányítási rendszere (ISMS - Information Security Management System), a legjobb gyakorlatok kézikönyve
ISO 27000 sorozat <- BS7799	Az informatikai biztonság irányítási rendszere (ISMS - Information Security Management System) a legjobb gyakorlatokat foglalja össze; **ISO 27017 Security Controls for Cloud Services**
ISO 20000 <- ITIL (Information Technology Infrastructure Library)	Az IT folyamatok kezelése, az információs technológia infrastruktúrájára, az IT fejlesztésére és működtetésére vonatkozó elvek és technikák gyűjteménye
SAS 70 (Statement on Auditing Standards)	Auditing szabvány, Bemutatja, hogy egy független könyvvizsgáló és auditáló cég hogyan ellenőrzi az IT szolgáltatók irányítását és az ehhez tartozó folyamatokat
ISACA (Information System Audit and Control Association)	Tanúsítványok • Certified Information Systems Auditor (CISA) • Certified in Risk and Information Systems and Control • Certified in the Governance of Enterprise IT • Certified Information Security Manager Az IT szakmai minőségbiztosítással foglalkozó szakemberek részére általában előírás e tanúsítások egyikének vagy többnek a megléte

SZABVÁNY / AJÁNLÁS / TANÚSÍTÁS	TARTALOM
IT TERVEZÉS MEGFELELŐSÉGE (IT Architecture compliance)	Az IT-re alkalmazandó megfelelési szabványok és keretek, pl. ISO 27001 információbiztonsági szabvány vagy COBIT 2019. Az ISO 27001 szabvány egy adott szervezeten belüli információbiztonsági rendszer létrehozására, megvalósítására, karbantartására és folyamatos javítására vonatkozóan határoz meg követelményeket.
GDPR (General Data Protection Regulation)	Az uniós szintű adatvédelmi törvény a magánszemélyek személyes adatainak magánszemély, vállalkozás vagy szervezet által történő kezelését szabályozza
Legfontosabb ipari automatizálási (IOT) szabványok és protokollok	• Programmable controllers: IEC 61131 • Generic model for distributed systems: IEC 61499 • Digital data communications for measurement and control – • Fieldbus for use in industrial control systems: IEC 61158/IEC 61784 • Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems: IEC 61508 • Functional safety - Safety instrumented systems for the process industry sector: IEC 61511 • machinery specific implementation of IEC/EN 61508: IEC 62061 • UL/CSA 61010-1 (Process Control Equipment) or UL/CSA/IEC 61010-2-201 (Programmable Logic Controllers) in conjunction with 61010-1 • CSA C22.2 Number 142 • UL 508 reg: E75310

9.7.7 Hazai informatikai szabványok, ajánlások

A nemzeti szabványosításról szóló 1995. évi XXVIII. törvény 4.§ (1) meghatározta, hogy: „A szabvány elismert szervezet által alkotott vagy jóváhagyott, közmegegyezéssel elfogadott olyan műszaki (technikai) dokumentum, amely tevékenységre vagy azok eredményére vonatkozik, és olyan általános és ismételten alkalmazható szabályokat, útmutatókat vagy jellemzőket tartalmaz, amelyek alkalmazásával a rendező hatás az adott feltételek között a legkedvezőbb." A hazai informatikát is érintő technológiai szabványokat, irányelveket, ajánlásokat a **Magyar Szabványügyi Testület** (MSZT) kezeli, ami közhasznú tevékenységet végző köztestületként működik. A testület a hazai szabványokat MSZ-szabványként adja ki, a Szabványügyi Közlönyben. Az egyes szabványok a honlapjukról online is megvásárolhatók. A **szabványok betartása önkéntes.** https://ugyintezes.mszt.hu/ Az adatbázisukban **több mint 1600 az informatikát**, a különféle szakterületeken alkalmazott információs rendszerek technológiáit, biztonsági előírásait, követelményeit leíró **MSZ szabvány** szerepel. Lásd pl.: https://n9.cl/hds86

Az informatikát érintő szabványok, ajánlások közül kiemelkedően fontos az **informatikai biztonságot** érintő szabványok és ajánlások ismerete. A magyar informatikai biztonsági szabványok a **HUNCERT** csoport honlapján egy összefoglaló táblázatban is megtalálhatóak: https://www.cert.hu/magyar-informatikai-biztonsagi-szabvanyok

SZABVÁNYOK / AJÁNLÁSOK	TARTALOM/FORRÁS
Informatikát érintő MSZ: szabványok	Magyar Szabványügyi Testület: https://ugyintezes.mszt.hu/
ezen belül a magyar informatikai biztonsági szabványok	HUNCERT csoport: https://www.cert.hu/magyar-informatikai-biztonsagi-szabvanyok

9.7.7.1 Jogi szabályozási környezet

Az IT tervezői feladatok ellátásához nemcsak a technológiai szabványok ajánlások, hanem az adott – pl. az ipari, kutatási, egészségügyi, közlekedési, közszolgáltatási, pénzügyi, piaci-kereskedelmi, stb.- szektorban működtetett informatikai rendszerekre, szolgáltatásokra vonatkozó érvényes jogszabályi előírások ismerete is szükséges. Ezek ismertetése, és felsorolása ebben a dokumentumban ugyancsak nem kezelhető, ezért itt egy olyan iránymutatást kívánunk adni a tervezőknek, hogy hol keresse az egy-egy szektorra, illetve adott szolgáltatás típusra vonatkozó jogszabályi előírásokat.

A jogszabályi környezet ismertetése a jelen dokumentum kiadásának időpontjában hatályos jogszabályokat ismerteti, így ezek tájékoztatási jellegű információk. Felhívjuk a tervezők figyelmét, hogy minden esetben a tervezés előkészítési fázisában ellenőrizzék az adott szakterületre érvényes, akkor hatályos jogszabályi és szabályozási környezetet és a technológiai ajánlásokat. Ez az útmutató abban kíván segítséget nyújtani, hogy áttekintést ad arról, mire kell figyelemmel lenni és milyen információt hol találhatnak meg. Ennek megfelelően a linkek és azok tartalma változhat, ebben az esetben az adott szerv vagy jogutódja honlapján kell a keresett információt kikeresni.

9.7.7.2 Legfontosabb EU rendeletek

MEGNEVEZÉS	LEÍRÁS	TARTALOM/FORRÁS
EU: eIDAS-rendelet	Az EU belső piacain történő elektronikus tranzakciókhoz kapcsolódó elektronikus azonosításról és bizalmi szolgáltatásokról	https://digital-strategy.ec.europa.eu/hu/policies/eidas-regulation; https://eur-lex.europa.eu/legal-content/HU/TXT/HTML/?uri=CELEX:32014R0910&from=HU
GDPR: EU általános adatvédelmi rendelete	a természetes személyeknek a személyes adatok kezelése tekintetében történő védelméről és az ilyen adatok szabad áramlásáról, valamint a 95/46/EK irányelv hatályon kívül helyezéséről	https://net.jogtar.hu/jogszabaly?docid=a1600679.eup

9.7.7.3 Fontos hazai informatikai jogszabályok

MEGNEVEZÉS	LEÍRÁS	TARTALOM/FORRÁS
Állami szervekre vonatkozó jogszabályok	Állami szervek működésére nyilvántartásaikra, adatbázisaikra vonatkozó jogszabályok gyűjteménye kereshető ki a kormány honlapján. az adott minisztérium hatáskörébe eső intézményre vonatkozó előírások	https://2010-2014.kormany.hu/hu/dok?type=311#!DocumentBrowse
Közszolgáltatásokra vonatkozó szabályozások listái	A hírközlési és elektronikus közszolgáltatási rendszerek kiépítésére, és szolgáltatásnyújtásra vonatkozó legfontosabb jogszabályok listái többek között a következő szolgáltatók/hivatalok honlapjain érhetően el: – NISZ Zrt; – Belügyminisztérium> Nyilvántartó; – NMHH: hírközlés, azonosító gazdálkodás, frekvencia, média felügyelet, építésügy, filmkészítés, posta, elektronikus szolgáltatások; – Kormányhivatalok; – DMÜ; Elektronikus Ügyintézési Felügyelet;	https://nisz.hu/jogszabalyok; https://www.nyilvantarto.hu/archiv_honlap/kozos/index.php?k=jogszabalyok_hu (archív oldal) https://www.nyilvantarto.hu/hu/adatszolgaltatas_szemelyi#jog https://nmhh.hu/tart/kereses?HNDTYPE=SEARCH&name=doc&page=1&fac_content_type=jogszabaly_rolunk https://n9.cl/gaf2yw https://www.dmu.gov.hu/cikkek/bemutatkozas-dmu https://euf.gov.hu/

MEGNEVEZÉS	LEÍRÁS	TARTALOM/FORRÁS
elektronikus ügyintézést érintő jogszabályok	2015. évi CCXXII. törvény az elektronikus ügyintézés és a bizalmi szolgáltatások általános szabályairól; 451/2016. (XII. 19.) Korm. rendelet az elektronikus ügyintézés részletszabályairól Elektronikus Ügyintézési Felügyelet	https://net.jogtar.hu/jogszabaly?docid=a1500222.tv; https://net.jogtar.hu/jogszabaly?docid=a1600451.kor https://euf.gov.hu/Search?q=jogszab%C3%A1ly&culture=hu-HU

MEGNEVEZÉS	LEÍRÁS	TARTALOM/FORRÁS
Információ és informatikai biztonságot érintő jogszabályok	Közigazgatási hatóságokra általánosan érvényes informatikai biztonságot is érintő jogszabályok elérhetőek: NKI: Nemzeti Kibervédelmi Intézet NEAK: Nemzeti Egészségbiztosítási Alapkezelő Szabályozott Tevékenységek Felügyeleti Hatósága (SZTFH): Kiberbiztonság felügyelet	https://nki.gov.hu/hatosag/tartalom/vonatkozo-jogszabalyok/ https://www.neak.gov.hu/felso_menu/lakossagnak/adatvedelem/elektronikus_informaciobiztonsag#Jogszab%C3%A1lyok https://sztfh.hu/tevekenysegek/kiberbiztonsag-felugyelete/ Magyar Közlöny 2023.május 15 hétfő 72. szám: – 2023. évi XXIII. törvény a kiberbiztonsági tanúsításról és a kiberbiztonsági felügyeletről – A Szabályozott Tevékenységek Felügyeleti Hatósága elnökének 10/2023. (V. 15.) SZTFH rendelete az információs és kommunikációs technológiák kiberbiztonsági tanúsításáról
iratkezelés szabályozása	A közszolgáltatásokat nyújtó szerveknél használható Tanúsított iratkezelő programok listája; A közfeladatot ellátó szervek és tartósan állami tulajdonban lévő gazdasági társaságok iratkezelésére vonatkozó jogszabályok	https://2010-2014.kormany.hu/hu/kozigazgatasi-es-igazsagugyi-miniszterium/kozigazgatasi-allamtitkarsag/hirek/iratkezelesi-felugyelet https://mnl.gov.hu/mnl/keml/iratkezeles

MEGNEVEZÉS	LEÍRÁS	TARTALOM/FORRÁS
Főbb szakági jogszabályok	2003. évi C. törvény az elektronikus hírközlésről; 2001. évi CVIII. törvény az elektronikus kereskedelmi szolgáltatások, valamint az információs társadalommal összefüggő szolgáltatások egyes kérdéseiről (Ekertv.) 2005. évi CLXXXIII. törvény a vasúti közlekedésről;	https://net.jogtar.hu/jogszabaly?docid=a0100108.tv https://net.jogtar.hu/jogszabaly?docid=a0100108.tv https://net.jogtar.hu/jogszabaly?docid=a0500183.tv
Főbb szakági jogszabályok	Egészségügyi informatikát érintő jogszabályok Pénzügyi biztosítási területet érintő jogszabályok	https://e-egeszsegugy.gov.hu/jogszabalyok; https://net.jogtar.hu/jogszabaly?docid=a2200029.kor; https://kancellaria.unideb.hu/sites/default/files/2021-08/jogszabalyok.pdf https://www.mnb.hu/letoltes/pszafhu-utmut-107cobit.pdf https://www.mnb.hu/felugyelet/szabalyozas/felugyeleti-szabalyozo-eszkozok/ajanlasok

9.7.8 Mérnök Kamara IT Tervező szakmagyakorlási feltételei

9.7.8.1 Az IT Tervező szakmagyakorlási jogszabályok

1. A **1997. évi LXXVIII. törvény** az épített környezet alakításáról és védelméről **felhatalmazza a Mérnök Kamarát a szakmagyakorlási jogok kiadására**: (https://net.jogtar.hu/jogszabaly?docid=99700078.tv)
2. A Kormány **618/2021. (XI. 8.) Korm. rendelete** az építésügyi és az építésüggyel összefüggő szakmagyakorlási tevékenységekről szóló **266/2013. (VII. 11.) Korm. rendelet** módosításáról **meghatározza azokat a szakmákat** és **szakterületeket**, ahol a Mérnök Kamara **szakmagyakorlási jogosultságokat adhat ki** és előírja szakmagyakorlási tevékenységekkel kapcsolatos kamarai **feladatokat** és **eljárásrendet**.
   i. **266/2013. (VII. 11.) Korm** rendelet **1. számú melléklet, II. Szakma, Építészeti-műszaki tervezés táblázat 18. tétele** meghatározza a **szakmagyakorlási** jogosultságokhoz szükséges képesítési **követelményeket,** szakmai **gyakorlati időket,** továbbá azokat a **feladatokat**, amelyeket az adott szakterületi jogosultsággal lehet végezni. (https://njt.hu/jogszabaly/2021-618-20-22;
   ii. Ugyanezen jogszabályi melléklet meghatározza a szabályozás **érvényességi hatókörét** abban, hogy a Kamara ezt a **szakmagyakorlási jogosultságot** a **létfontosságú rendszerek és létesítmények** azonosításáról és kijelöléséről és védelméről szóló törvény alá tartozó létesítményekkel **kapcsolatos informatikai beruházások informatikai terveinek készítésére** és **ellenőrzésére** vonatkozik. (https://net.jogtar.hu/jogszabaly?docid=a1300266.kor)
3. A **létfontosságú rendszerek és létesítmények** azonosításáról és kijelöléséről és védelméről szóló **2012. évi CLXVI. Törvény** 1. számú melléklete **meghatározza,** hogy melyek azok az **ágazatok**, amelyekbe tartozó **szolgáltatás, eszköz, létesítmény** vagy **rendszer** mely **rendszereleme,** továbbá az azok által nyújtott **szolgáltatások** azok, amelyek **elengedhetetlenek a létfontosságú** társadalmi feladatok ellátásához – így különösen az egészségügyhöz, a lakosság személy- és vagyonbiztonságához, a gazdasági és szociális közszolgáltatások biztosításához, az ország honvédelméhez, – és amelynek kiesése e feladatok folyamatos ellátásának hiánya miatt jelentős következményekkel járna. (https://net.jogtar.hu/jogszabaly?docid=a1200166.tv)

a. A törvény által **kijelölt létfontosságú rendszerelemek, amelyeknél a Kamara által szakmagyakorlási jogosultságot szerzett informatikai tervezőt kell igénybe** venni, a következő táblázatban láthatóak:

ÁGAZAT		ALÁGAZAT
2.	Energia	villamosenergia-rendszer létesítményei (kivéve az atomerőmű nukleáris biztonságára és sugárvédelmére, fizikai védelmére, valamint biztosítéki felügyeletére vonatkozó szabályozás hatálya alá tartozó rendszerek és rendszerelemek)
3.		kőolajipar
4.		földgázipar
5.		távhő
6.	Közlekedés	közúti közlekedés
7.		vasúti közlekedés
8.		légi közlekedés
9.		vízi közlekedés
10.		logisztikai központok
11.	Agrárgazdaság	mezőgazdaság
12.		élelmiszeripar
13.		elosztó hálózatok
14.	Egészségügy	aktív fekvőbeteg-ellátás, és a működtetéséhez szükséges szolgáltatások
15.		mentésirányítás
16.		egészségügyi tartalékok és vérkészletek
17.		magas biztonsági szintű biológiai laboratóriumok
18.		gyógyszer-nagykereskedelem
19.	Társadalombiztosítás	társadalombiztosítási ellátások igénybevételéhez kapcsolódó informatikai rendszerek és nyilvántartások
20.	Pénzügy	pénzügyi eszközök kereskedelmi, fizetési, valamint klíring- és elszámolási infrastruktúrái és rendszerei
21.		bank- és hitelintézeti biztonság
22.		készpénzellátás
23.	Infokommunikációs technológiák	internet-hozzáférési szolgáltatás és internet-infrastruktúra
24.		elektronikus hírközlési szolgáltatások, elektronikus hírközlő hálózatok
25.		műsorszórás

ÁGAZAT		ALÁGAZAT
26.		postai szolgáltatások
26a		meteorológiai infrastruktúra
27.		kormányzati elektronikus információs rendszerek
28.	Víz	ivóvíz-szolgáltatás
29.		felszíni és felszín alatti vizek minőségének ellenőrzése
30.		szennyvízelvezetés és -tisztítás
31.		vízbázisok védelme
32.		árvízi védművek, gátak
33.	Honvédelem	honvédelmi rendszerek és létesítmények
34.	Közbiztonság-védelem	rendvédelmi szervek infrastruktúrái

9.7.8.2 Létfontosságú rendszerek, létesítmények jogszabályai

Az nki.gov.hu oldalán található kimutatás szerint:

- Magyarország hálózati és információs rendszerek biztonságára vonatkozó Stratégiája [HUN / ENG]
- 2012. évi CLXVI. törvény a létfontosságú rendszerek és létesítmények azonosításáról, kijelöléséről és védelméről (Lrtv.) [HUN / ENG]
- 2013. évi L. törvény az állami és önkormányzati szervek elektronikus információbiztonságáról (Ibtv) [HUN / ENG]
- 65/2013. (III. 8.) Korm. rendelet a létfontosságú rendszerek és létesítmények azonosításáról, kijelöléséről és védelméről szóló 2012. évi CLXVI. törvény végrehajtásáról (Lrtv. vhr.) [HUN / ENG]
- 541/2013. (XII. 30.) Korm. rendelet a létfontosságú vízgazdálkodási rendszerelemek és vízilétesítmények azonosításáról, kijelöléséről és védelméről [HUN / ENG]
- 246/2015. (IX. 8.) Korm. rendelet az egészségügyi létfontosságú rendszerek és létesítmények azonosításáról, kijelöléséről és védelméről [HUN / ENG]
- 330/2015. (XI. 10.) Korm. rendelet a pénzügyi ágazathoz tartozó létfontosságú rendszerek és létesítmények azonosításáról, kijelöléséről és védelméről [HUN / ENG]
- 249/2017. (IX. 5.) Korm. rendelet az infokommunikációs technológiák ágazathoz kapcsolódó létfontosságú rendszerek és létesítmények azonosításáról, kijelöléséről és védelméről [HUN / ENG]
- 161/2019. (VII. 4.) Korm. rendelet a közlekedési létfontosságú rendszerek és létesítmények azonosításáról, kijelöléséről és védelméről [HUN / ENG]
- 374/2020. (VII. 30.) Korm. rendelet az energetikai létfontosságú rendszerek és létesítmények azonosításáról, kijelöléséről és védelméről [HUN / ENG]
- 187/2015. (VII. 13.) Korm. rendelet az elektronikus információs rendszerek biztonsági felügyeletét ellátó hatóságok, valamint az információbiztonsági felügyelő feladat- és hatásköréről, továbbá a zárt célú elektronikus információs rendszerek meghatározásáról [HUN / ENG]
- 270/2018. (XII. 20.) Korm. rendelet az információs társadalommal összefüggő szolgáltatások elektronikus információbiztonságának felügyeletéről és a biztonsági eseményekkel kapcsolatos eljárásrendről [HUN / ENG]
- 271/2018. (XII. 20.) Korm. rendelet az eseménykezelő központok feladat- és hatásköréről, valamint a biztonsági események kezelésének és műszaki vizsgálatának, továbbá a sérülékenységvizsgálat lefolytatásának szabályairól [HUN / ENG]

– 41/2015. (VII. 15.) BM rendelet az állami és önkormányzati szervek elektronikus információbiztonságáról szóló 2013. évi L. törvényben meghatározott technológiai biztonsági, valamint a biztonságos információs eszközökre, termékekre, továbbá a biztonsági osztályba és biztonsági szintbe sorolásra vonatkozó követelményekről [HUN / ENG]

9.7.8.3 MMK Informatikai engedélyköteles szakterületek

MMK - IT terület Titulus	Típus	MMK Hatáskör
„IT - Informatikai Tervező”	jogosultság	MMK tagsághoz kötött, a „266/2013. (VII. 11.) Korm. rendelet az építésügyi és az építésüggyel összefüggő szakmagyakorlási tevékenységekről” alapján
„IN-PE - Informatikai Projekt Ellenőr”	jogosultság	nyilvántartását a MMK végzi és ellenőrzi
„IPF - Informatikai Projekt Felügyelő”	tanúsítvány	MMK kamarai tagsághoz kötött

9.7.8.4 IT Tervező szakmagyakorlás feltételei

Megszerzésének feltételeit a 266/2013-as kormány rendelet 1. melléklete szabályozza.

Szakmagyakorlási terület megnevezése		Feladatok, amelyeket az adott szakterületi jogosultsággal lehet végezni	Képesítési követelmény vagy ezzel egyenértékű szakképzettség	Szakmai gyakorlati idő
Informatikai tervezési szakterület	IT	A létfontosságú rendszerek és létesítmények azonosításáról és kijelöléséről és védelméről szóló törvény (2012. évi CLXVI. Törvény) alá tartozó létesítményekkel kapcsolatos informatikai beruházások informatikai terveinek készítése és ellenőrzése.	okleveles villamosmérnök, okleveles mérnökinformatikus, okleveles műszaki informatikus, okleveles programtervező informatikus, okleveles gazdaságinformatikus, okleveles alkalmazott matematikus, okleveles programtervező matematikus	3 év
			villamosmérnök, mérnökinformatikus, programtervező informatikus, gazdaságinformatikus, üzemmérnök-informatikus (BPROF), programtervező matematikus, programozó matematikus	5 év

9.7.8.5 Az MMK Informatikai Tervezői jogosultság igénylés

Az alábbiakban ismertetett folyamat és linkek a jelenleg (2023.06-10) hatályos eljárásrendet tükrözik. Amennyiben a kamarai szabályzat a későbbiekben módosul, akkor a hatályos „Tervezőkre" vonatkozó eljárásrendet a mindenkori kamarai honlapon kell megkeresni.

A jelenlegi jogosultság igénylés folyamata

Eljárásrend típus	Szabályozás
Jogosultság kiadás	https://www.mmk.hu/informaciok/dokumentumok/szabalyzatok/Hatalyos-szabalyzatok/
Jogosultság fenntartás	https://www.mmk.hu/informaciok/dokumentumok/szabalyzatok/Hatalyos-szabalyzatok/

Informatikai Tervezői jogosultság igénylése	• Magyar Mérnök Kamarai tagság megszerzése (http://mmk.hu/szervezet/megyei_kamarak;
	• http://www.bpmk.hu/index.php/2014-06-23-13-34-37 (Kamarai jelentkezesi lap_2018.doc).
	• A kérelem melléklete a szakmai önéletrajz, a végzettség és a regisztrálási díj befizetés igazolása
	• A kérelemben a szakmai tagozat megjelölése: **Hírközlési és Informatikai Tagozat**
	• Szakterület megjelölése: **Informatikai Tervező**
	• Informatikai Tervezői jogosultság igénylése.
	• Végzettség és szakmai kompetenciák felmérése és a gyakorlat igazolása
	  ○ Szakmai kompetenciák felmérése
	  ▪ képzettségi megfelelés vizsgálat
	  ▪ szakmai gyakorlat felmérése beszámolóval
	  ▪ https://www.mmk.hu/informaciok/dokumentumok/szabalyzatok/Hatalyos-szabalyzatok oldal szabályzatai szerint

Eljárásrend típus	Szabályozás
Jogosultsági vizsga	• A jogosultsági és beszámoló vizsgák (továbbiakban vizsgák) tartalmát és lebonyolítását az építésüggyel összefüggő szakmagyakorlási tevékenységekről szóló 266/2013. (VII. 11.) Korm. rendelet szabályozza. • Jogosultsági vizsga tájékoztatója: https://www.mmk.hu/ugyintezes/vizsga
Szakmai szintű nyilvántartás igénylése	• Megyei kamarákhoz jelentkezés után: http://www.bpmk.hu/index.php/jogosultsagi-uegyek
Kötelező továbbképzések	• Kötelező továbbképzési naptár: https://www.mmk.hu/kepzesek/szakmai-tovabbkepzes • Az MMK eljárásrendje alapján: https://www.mmk.hu/informaciok/dokumentumok/szabalyzatok/Hatalyos-szabalyzatok
Szakmagyakorlás	• https://www.mmk.hu/informaciok/dokumentumok/szabalyzatok/Hatalyos-szabalyzatok -szakmagyakorlási szabályzatai alapján

9.7.8.6 Lebonyolítási folyamat ismertetése

Az alábbiakban bemutatott folyamat a jelenleg érvényes eljárás rendet tükrözi, amit a könnyebb átláthatóság kedvéért összefoglalva kigyűjtöttünk.

1. **Végzettség és szakmai kompetenciák felmérése és a gyakorlat igazolása:**
   a. Külföldön szerzett oklevél végzettség csak honosítás után fogadható el.
   b. Szakmai önéletrajz tartalma az iskolai tanulmányok, fontosabb továbbképzések, munkahelyek, munkahelyen végzett jelentősebb tevékenységek, valamint a munkahelytől függetlenül végzett szakmai tevékenységek, pl. szakmai egyesületi tagság, magántervezés, szakértés esetleg korábbi jogosultságok ismertetése.
   c. Az egyetemi/főiskolai végzettség megszerzése óta eltelt időszak alatt végzett szakmai gyakorlatként az informatikai tervezési, informatikai rendszer kivitelezési, beruházási műszaki/informatikai, közigazgatásban műszaki/informatikai, felsőoktatási intézménynél műszaki/informatikai szaktárgyat oktató tevékenységet lehet figyelembe venni.
   d. Nem tekinthető az Informatikai Tervezői gyakorlat részének: pl. a kereskedelmi, a nem műszaki/informatikai államigazgatási, a művészeti, a nem mérnöki/informatikai katonai, a vagyonvédelmi, stb. tevékenység.
   e. A részben mérnöki/informatikai tervezési, részben nem mérnöki/informatikai tervezési tevékenységnél (pl. a munkaidőn kívül végzett tevékenység esetén) az időarányos részt lehet figyelembe venni.
   f. A belépésre irányuló kérelemben meg kell jelölni, hogy a kamara HIT szakmai tagozathoz szeretne elsődlegesen tartozni, illetve további tagozatok is választhatók a szakmagyakorlási szakterületeknek megfelelően.
   g. A Magyar Mérnöki Kamara szervezeti rendszerében a szakmai tagozatokon belül Szakosztályok működnek, amelyek az egyes területek cél szerinti szűkebb csoportját jelölik, ilyen pl. a Hírközlési és Informatikai Tagozaton (HIT) belül működő Informatikai Szakosztály is.
   h. A kamarába való felvételről a területi kamara elnöksége a törvényi feltételek igazolása után 30 napon belül határoz, és azt írásban megküldi a kérelmezőnek.
   i. A tagfelvétel elutasítását indokolni kell. A felvételt megtagadó határozat ellen a kérelmező a kézbesítéstől számított 15 napon belül az országos kamarához fellebbezhet. A fellebbezést 30 napon belül el kell bírálni.
   j. Belépéssel a kamara tagja magára nézve kötelező erővel elfogadja a Magyar Mérnöki Kamara szabályzatait, különösen az Etikai- és Fegyelmi Szabályzatot, mely „garanciát jelent mind a mérnökök egymással szembeni viszonyában, mind a mérnök és a megrendelő közötti kapcsolatokban.”
   k. Az MMK tagja éves tagdíjat fizet, amelynek mértéke attól függ, hogy jogosultsággal vagy tanúsítvánnyal rendelkezik vagy nem, illetve egyéb körülmények (életkor, GYED, tiszteletbeli cím, örökös tagság) mérséklik azt. Kamarai tagsággal nem rendelkező nyilvántartottak is éves díjat fizetnek. Egy adott naptári év első félévében belépők a tagdíj teljes összegét, míg a második félévben belépők a tagdíj 50 %-át fizetik.

2. Szakmai szintű nyilvántartás igénylése
   a. Felvételkor a mérnöki tervezői és szakértői tevékenységek végzéséhez hasonlóan az Informatikai Tervezői szakmai szintű nyilvántartáshoz is külön űrlap kitöltése szükséges a lakóhelyi megyei kamara felé.Pl:
   b. MMK tag szakmai szintű nyilvántartását nem csak felvételkor, hanem később, bármikor kérheti. A kérelem melléklete a szakmai önéletrajz, a végzettség és a szakmai gyakorlat, valamint az eljárási díj befizetés igazolásai.
   c. A kérelmet az MMK-hoz kell beadni. A kérelem vizsgálat nélkül elutasításra kerül, ha kérelmező a kérelmét hiányosan adta be.

3. Előfeltételek vizsgálata
   a. A beérkezett kérelem a mellékletekkel együtt 3 munkanapon belül az illetékes szakmai tagozat szakértői testületéhez kerül. A szakértői testület 30 napon belül szakmai döntést hoz az előfeltételek teljesüléséről, jogosultsági eljárásban a vizsgára jelentkezés elfogadásáról vagy elutasításáról. A szakértői testület az előfeltételek teljesülésének vizsgálatakor kérheti, hogy a kérelmező szóbeli beszámolóval is igazolja a szakmai gyakorlatát. Ebben az esetben az eljárás e szakaszának 30 napos határideje a szóbeli beszámoló lebonyolításának idejével meghosszabbodik.
   b. Ha a szakértői testület a jelentkezést elutasítja, döntését indokolni köteles.
   c. Az Informatikai Tervezői jogosultság megadása iránti kérelem, amennyiben az indokolásban foglaltak szerint az elutasítás alapjául szolgáló körülmények megváltozása ezt lehetővé teszi, ismételten benyújtható.

3.1. Képzettségi megfelelés vizsgálat

   a. A szakirányú végzettséget a leckekönyvvel és a kérelmezett szakterülethez kapcsolódó további oklevelek, bizonyítványok másolatával kell igazolni.
   b. A kamarai nyilvántartásoknál a képzettségi megfelelés szempontrendszerének kidolgozása a felsőfokú oktatási intézmények kreditrendszere alapján történik.

3.2. Szakmai gyakorlat felmérése

   a. A szakmai gyakorlat és a referencia igazolásaként csak közokirat vagy teljes bizonyító erejű magánokirat, megrendelői, valamint megbízói nyilatkozat fogadható el, amely tartalmazza a gyakorlat időtartamát, a végzett munkák megnevezését, a folytatott tevékenység leírását.
   b. A megfelelés vizsgálat szempontjai:

      • A referenciaigazolás jogszabályoknak történő formai megfelelősége.

      o Kamarai tag igazolása esetén Informatikai Tervezői jogosultsággal/tanúsítvánnyal rendelkező szakmagyakorló, mentor igazolta a referenciát

      • A referencia tevékenységeknek tartalmi megfelelősége, a kérelmezett szakmai előírásokkal történő megfelelősége.

      • Az előírt szakmai gyakorlati időt el kell érni az elfogadott referenciák évenkénti bontásának.

      • Benyújtott és elfogadott dokumentumok, adatok alapján igazolható az elvárt jártasság, felelős munkavégzés

3.3. Szóbeli beszámoló

a. A szakértői testület döntése lehet, hogy kérelmező szakmai gyakorlatát – a szakmai feltételek teljesítésén túl - szóbeli beszámolóval is igazolja.
b. A szakmai gyakorlatok, informatikai tárgyú tervek kamarai választott legalább 3 tagú vizsgabizottság előtt történő bemutatása, visszacsatolással az esetleges hiányosságokról.
c. A vizsga időtartalma 30 perc.
d. Szóbeli beszámoló célja az igazolásként bemutatott gyakorlati idő feltárása, projektek bemutatása, vizsgabizottság kérdéseinek megválaszolása annak érdekében, hogy a jelölt gyakorlatból szerzett készségek és tudás mélységének megfelelősége bizonyítottá váljék.
e. A szükséges tapasztalat a projektek alkalmas, kitakart (cenzúrázott) terveinek bemutatásával igazolható. Titkosított projektet egyedi esetként kezeljük.
f. Részletes szakmai szempontokat a III. kötet tartalmazza.
g. Az eljárásban közreműködő szakértői testület állásfoglalásáról, döntéséről jegyzőkönyvet készít, melynek a szokásos formai elemeken túl tartalmaznia kell:

   • az ügyre vonatkozó lényeges nyilatkozatokat és megállapításokat,

   • az esetleges eljárási cselekmények során tapasztalt, az ügy eldöntése szempontjából lényeges körülményeket és megállapításokat,

   • a döntést,

   • és elutasítás esetén annak indokolását.

4. Írásbeli jogosultsági vizsga

a. A jogosultsági vizsga általános része alól a szakmagyakorló felmentést kaphat, ha más jogosultság megszerzéséhez ilyen vizsgát már tett, vagy rendelkezik olyan vizsgával, amelynek követelménye 80%-ban megegyezik az általános rész követelményrendszerével. A felmentésről szóló döntést a Beszámoló Vizsga Szakértő Testület (BVSZT) hozza meg, illetve ő állapíthatja meg a mentesülés tényét.
b. A felmentési kérelmet az MMK Főtitkárának címezve kell benyújtani (beszamolo@mmk.hu).
c. A felmentési kérelem és a többi nyomtatvány letölthető: https://www.mmk.hu/ugyintezes/vizsga.
d. A jogosultsági vizsga szakterületi része alól felmentés nem adható.

4.1. Vizsgára jelentkezés

a. A jogosultsági vizsgák időpontjait az MMK online naptárában a http://mmk.hu/e-mernok/vizsganaptar honlapon előre meghirdeti.
b. Egy-egy vizsgaalkalom minimális létszáma 10 fő, maximális létszáma 20 fő. Az általános és a szakterületi vizsgákra külön-külön kell jelentkezni. Egy vizsganapon mind az általános, mind a szakterületi vizsga teljesíthető, de lehetőség van arra, hogy a két vizsgarészt a vizsgázó külön teljesítse.
c. A vizsgára elektronikusan, azaz a kamara honlapján elérhető jelentkezési lap elektronikus kitöltésével és az országos kamarának való megküldésével kell jelentkezni. Egy személy vizsgatípusonként csak egy időpontra jelentkezhet. Új időpontra csak akkor lehet jelentkezni, ha az előzőt a vizsgázó törölte vagy azon a vizsgakövetelményeket nem tudta teljesíteni. Amennyiben a létszám egy alkalomra betelik, úgy a rendszer arra az alkalomra több jelentkezést nem fogad el.
d. A vizsgára való jelentkezés csak abban az esetben lehetséges, ha a vizsgázó bejelentkezett az e-Mérnök rendszerbe, ennek menete: https://www.mmk.hu/e-mernok/jelentkezoknek
e. A vizsgázó köteles elektronikusan értesíteni az MMK-t (beszamolo@mmk.hu), amennyiben a vizsgán nem tud részt venni. Ha a vizsgára jelentkező legkésőbb a vizsga napját megelőző második munkanapig nem tesz eleget értesítési kötelezettségének, akkor a befizetett vizsgadíj összegéből az adminisztrációs költségek levonásra kerülnek.

4.2. Vizsga tárgya

a. A jogosultsági vizsgákon az általános és a szakterületi vizsgarészen egyaránt 10-10 kérdést kell megválaszolni. Mind az általános, mind a szakterületi vizsgarészben 5 kérdés a legalapvetőbb ismeretekre, 5 kérdés pedig az összetettebb ismeretekre vonatkozik. A tesztkérdések között van eldöntendő, kiválasztó, feleletválasztó és kiegészítő típusú kérdéssor is.
b. A kérdések megválaszolására a vizsgázók a rendelkezésre álló szakirodalomból, jogszabályokból és szabványokból önállóan is felkészülhetnek. A vizsgára való közvetlen felkészülést segítendő kéziratok állnak a vizsgázok rendelkezésére, melyek az MMK honlapjáról letölthetők: http://mmk.hu/szolgaltatasok/vizsga/felkeszulesi-segedletek.
c. Az aktuális vizsgán feltett tesztkérdéseket a BVSZT a fenti kérdésbank alapján állítja össze, de ezek nem egyeznek meg a kérdésbankban található kérdésekkel, ugyanakkor a jellegüket a fenti honlapra feltett példák érzékeltetik.
d. A vizsgán jelenléti ív készül és a helyszínen a vizsga megkezdése előtt a vizsgázóknak igazolniuk kell személyazonosságukat fényképes személyazonosító igazolvánnyal (személyazonosító igazolvány vagy útlevél vagy gépjárművezetői engedély).
e. A jogosultsági vizsga alapvető formája a számítógépes tesztvizsga, és a számítógépeket az MMK biztosítja.
f. Az általános kamarai gyakorlatnak megfelelően a 266/2013. (VII. 11.) Korm. rendelet 39. § (3) bekezdése vizsga során a segédeszközök használatát kifejezetten megtiltja.
g. Erről a vizsgázók a vizsga megkezdése előtt külön tájékoztatást is kapnak. Azt a vizsgázót, aki e szabályt megszegi, a vizsgabizottság kizárhatja a vizsgáról,

amely így érvénytelen vizsgának minősül. A vizsga szabályainak megszegése miatt érvénytelennek nyilvánított vizsgát követően a vizsgázó csak a vizsgadíj ismételt befizetését követően jelentkezhet új vizsgára.

h. A vizsgabizottság helyben, azonnal értékeli az írásbeli dolgozatokat.
i. Az általános rész és a szakterületi rész eredményét a BVSZT külön-külön értékeli. Minden tesztkérdés azonos súllyal számít. Csak a teljes körű válasz tekinthető helyesnek.
j. A részvizsga követelményeit az teljesíti, aki az írásbeli kérdések 70%-át helyesen válaszolta meg.
k. Az 50-70% közötti írásbeli vizsga eredmény esetében a vizsgázó szóbeli vizsgát is tesz. A szóbeli vizsgán a vizsgázó további kérdés(eke)t kap.
l. Az 50% alatt teljesített vizsga eredménytelen, szóbeli beszámolóra a vizsgázó nem bocsátható, az írásbeli vizsgát ismételni kell.
m. A vizsga eredménytelen, ha a vizsgázó bármelyik részből nem megfelelő eredményt ért el. Újból vizsgáznia azonban csak azon részből szükséges, amely részből eredménytelen minősítést kapott.
n. A vizsgázó a saját dolgozatának értékelésébe a javítás után, kérésre betekinthet.
o. Az írásbeli értékelés befejezését követően a testület kihirdeti a vizsgázók eredményeit és a szóbeli beszámolóra kötelezettek névsorát, továbbá a szóbeli pontos időpontját.
p. A szóbeli vizsgát alapesetben az írásbeli után azonnal, kivételes esetben néhány napon belül kell megtartani.
q. A jogosultsági vizsga mindkét (általános és szakterületi) részét eredményesen teljesítők az eredményhirdetéseket követően megkapják a bizonyítványt.
r. A javító jogosultsági vizsga díja első alkalommal ingyenes, ezt követően a jogosultsági vizsga díjával azonos.

5. **Kötelező továbbképzések**
   a. Az Informatikai Tervező jogosultság érvényessége 5 évre szól, ha a szakember teljesíti az évente kötelező szakmai továbbképzés előírásait.
   b. A jogosultságot és a tanúsítványt kérelemre meg lehet hosszabbítani és ekkor a szakmai továbbképzés teljesítésén kívül a jogi továbbképzést is teljesíteni kell.
   c. Továbbképzések formája tesztvizsgával záruló, internet alapú távoktatás, vagy hagyományos, kontaktórás oktatás lehet.
   d. Jogi továbbképzés
   e. A jogi továbbképzést az MMK országosan egységes rendszer kereteiben biztosítja. A jogi továbbképzés keretében a képzéseket - a Kamarai Továbbképzési Testület felügyelete mellett - az erre a feladatra létrehozott Mérnöki Kamarai Tudásközpont szervezi.

9.7.8.7 Szakmai továbbképzés

a. A szakmai továbbképzés teljesítésének a feltétele, hogy a szakmagyakorló évente részt vegyen az Informatikai Tervező szakterülethez kapcsolódó összesen 6 * 45 perces szakmai továbbképzésen.
b. A továbbképzési szabályzat 8. § (3) bk szerint: „Amennyiben a szakmagyakorlónak több szakmai tagozat kompetenciájába tartozó szakmagyakorlási jogosultsága van, az öt év alatt legalább évente egy szakmai

továbbképzést köteles teljesíteni azzal, hogy valamennyi szakmagyakorlási jogosultságának megfelelő szakmai továbbképzést legalább egyszer köteles teljesíteni."

c. Az Informatikai Tervező szakterület oktatandó témáit a Híradástechnikai és Informatikai tagozat vagy a témakörben a tagozatot képviselő Informatikai Szakosztály javasolja.

9.7.8.8 Szakmagyakorlás

A kamara jelenlegi Szakmagyakorlási Ellenőrzési szabályzata (https://www.mmk.hu/informaciok/dokumentumok/szabalyzatok/Hatalyos-szabalyzatok) egyrészt építésügyi szemléletre korlátozódik, másrészt „kiterjed az engedélyhez és/vagy névjegyzékbe vételi kötelezettséghez kötött mérnöki tevékenységet folytatók körére".

10 Irodalomjegyzék

A sorozat keretében eddig megjelent kiadványok

1.	GÁBORI László Dr., BEINSCHRÓTH József Dr., NÓGRÁDI Gábor, RÁTKAY Tamás	Nagyméretű informatikai beruházásoknál (fejlesztéseknél) ajánlott szoftveroldali tervdokumentációk tartalmi elemeinek meghatározása (I. – II. kötet)
20.	DR. GÁBORI László, DR. BEINSCHRÓTH József, NÓGRÁDI Gábor, RÁTKAY Tamás	Informatikai Tervező szakmai minősítő rendszere (Informatikai szakmai terület illesztése a Mérnök Kamarai működési rendbe és rendszerekbe) I. kötet: Koncepció és modell II. kötet: Modell illesztése III. kötet: Tudástár
	DR. GÁBORI László, DR. MOLNÁR Bálint, NÓGRÁDI Gábor, RÁTKAY Tamás	Az Informatikai Tervező tervezési segédlete

11 A sorozat kiadványai

2017.

1.	NÉMETH András, MILÁVECZ Richárd	Iparban használatos vízminőségek
2.	SZILÁGYI Zsombor Dr, SZUNYOG István Dr.	Mérések a gáziparban
3.	BARNA Lajos Dr., EÖRDÖGHNÉ MIKLÓS Mária Dr., SZÁNTHÓ Zoltán, BALLA József Dr.	A biztonságos ívóvízellátás megteremtésének tervezési eszközei
4.	BORBÁS Lajos Dr.	Felépítés elvű (additív) gyártástechnológiák a gépészetben
5.	BERENCSI Miklós, BERECZKY Ákos, HORVÁTH László, KOVÁCS Gergely, MIHÁLFFY Krisztina	Kerékpárosbarát közlekedéstervezés
6.	TÜDŐS Tibor, VARJÚ György Dr., PETRI Kornél Dr., GÁBOR András	A csillagpontkezelés legújabb külföldi és hazai eredményei (Útmutató és tervezési segédlet)

7.	**GARBAI László Dr., JASPER Andor Dr., VÁRADI András**	**Fűtési és használati melegvíz-igények kockázati elvű méretezése példákkal**
8.	**KÁDI Ottó, DOHÁNY Máté, JÓZSA Bálint, LÁSZLÓ Csaba Tibor, JAKKEL Ottó**	**A közúti vasutak (villamos) tervezésével kapcsolatos kézikönyv**

9.	BLAZSOVSZKY László	A gázfogyasztó készülékek égéstermék elvezetésével kapcsolatos szabályozások hiányosságai és ellentmondásai
10.	CSORDÁS Szilveszter, FORGÁCS Lajos Dr., PÓLYA Endre ifj., RÉV Zoltán, UDVARDY Péter	Orvostechnológiai továbbképzés ismeretanyaga
11.	NÁDASDY Tamás, EGYHÁZY Zita, KOVÁCS Ákos Sándor, SZECSŐ Dániel Géza	A közúti biztonsági audit (KBA) jelentések elkészítésének alkalmazási segédlete – A közúti infrastruktúra közlekedésbiztonsági kezeléséről szóló jogszabályhoz és útügyi műszaki előíráshoz kapcsolódó értelmezési, kidolgozási és elfogadtatási javaslatrendszer
12.	SZILÁGYI Zsombor Dr., HORÁNSZKY Beáta	Földgáz kereskedelem (mérnöki segédlet)
13.	SZILÁGYI Zsombor Dr.	Az energiahordozók jövője – kőolaj, földgáz, megújulók

14.	S. VÍGH Judit, DOHÁNY Máté	Magános közlekedők baleseti súlyosságának csökkentése mobil applikáció segítségével
15.	BALIKÓ Sándor Dr., CSŰRÖK Tibor Dr., NOVÁK Dániel, ORBÁN Tibor, ZSEBIK Albin Dr.	Ötletlapok I. – Energiahatékonyság növelő ötletek egyszerű energetikai és gazdasági számításai
16.	DARABOS Zoltán, KOLTAI Henrik, SZABÓ Tamás, SZÁSZ Béla, VAJDA Sándor	Felvonók felújítása és átalakítása – Műszaki segédlet
17.	TÜDŐS Tibor, KRUPPA Attila	Alapozásföldelők új tervezési elvei és kivitelezési módszerei – Tervezési segédlet és kivitelezési útmutató
18.	FENYVESI Zsolt	Tűzvédelmi tervek tartalmi szabályainak átdolgozása
19.	GÁBORI László Dr., BEINSCHRÓTH József Dr., NÓGRÁDI Gábor, RÁTKAY Tamás	Nagyméretű informatikai beruházásoknál (fejlesztéseknél) ajánlott szoftveroldali tervdokumentációk tartalmi elemeinek meghatározása (I. – II. kötet)

20.	DIVÓS Ferenc Dr.	Az élő fák stabilitása – mérnöki megközelítés – Élő fák, mint teherhordó faszerkezetek
21.	KARÁCSONYI Zsolt Dr.	Faanyagok tartós szilárdsága
22.	BARNA Lajos Dr., ERDEI István, JASPER Andor Dr., TAKÁCS Gyula	Segédlet épületek csatorna-berendezéseinek tervezéséhez
23.	ANTÓK Péter István, FÜZÉR Ferenc, SÁRKÖZI András	Fényvezető kábelszakaszok műszaki-minőségi ajánlás gyűjteménye
24.	JANCSÓ Béla, KULCSÁR Alexandra Dr., NÉMETH Gábor, VÍMI Zoltán Dr., DÉRI Lajos, SZIMANDEL Dezső	Vízjogi engedélyezési eljárással kapcsolatos dokumentációk és engedélyeztetéssel kapcsolatos követelmények a 2018.01.01-én hatályba lépett 41/2017. (XII.29.) BM rendelet alapján
25.	TAKÁCS Bence Dr., SIKI Zoltán Dr., ÉGETŐ Csaba Dr., BÉNYI László	Mérnökgeodéziában alkalmazott alapponthálózatok – A jó gyakorlat bemutatása mintapéldákkal
26.	MÓCZÁR Balázs Dr., LAUFER Imre, TÓTH Gergő, WOLF Ákos	Korszerű támszerkezetek tervezése

27.	HALÁSZ Györgyné Dr., CSERVENYÁK Gábor, TUCZAI Attila, VIRÁG Zoltán	Különböző funkciójú épületek klímatechnikája II.
28.	KÁDI Ottó, JÓZSA Bálint	Kerékpáros balesetek létesítmények szerinti vizsgálata
29.	GARBAI László Dr., JASPER Andor Dr., PELLER József Bendegúz	Hőteljesítményátviteli tényező alkalmazása távhőrendszerek optimális szabályozásának modelljében
30.	GARBAI László Dr., SÁNTA Róbert Dr., JASPER Andor Dr.	A kompresszoros hőszivattyúk optimalizálása – Tervezés és üzemeltetés
31.	LADÁNYI Gábor Dr.	Diagnosztika a karbantartásban
32.	MÉSZÁROS János, MOLNÁR Tibor, RITZL András	KIÜRÍTÉSI ÉS MENEKÜLÉSI ÚTVONALBA ÉPÍTETT AJTÓK tervezési segédlet (2018)

33.	BLAZSOVSZKY László	Földgáz elosztóvezetékek üzemeltetése
34.	DR. SZILÁGYI Zsombor	A megújuló energiahordozók jövője Magyarországon
35.	FORGÁCS Lajos Dr., HAIDEGGER Tamás Dr., PÓLYA Endre ifj.	Új fejlesztések, innovatív megoldások az orvostechnológia terén
36.	VARRÓ Beáta, KIS András Dr.	Magyarországon előforduló, épületekbe beépített faanyagokat károsító gombák vizsgálata és azonosítása DNS diagnosztikával
37.	MANNINGER Marcell, SZEPESHÁZI Attila, SCHEURING Ferenc, MOLNÁR György	Munkatér határoló szerkezetek
38.	KORSÓS András, RÁDULY Zsolt	A közterületi és belterületi térfigyelő kamerarendszerek tervezési irányelvei
39.	GERGELY Edit, BEZEGH András Dr.	Módszertani útmutató az üvegházhatású gázok közvetlen és közvetett kibocsátásának számítására

40.	BEZEGH András Dr., BITE Pálné Dr., GERGELY Edit	Városi környezetvédelem (Fenntartható és okos városok)
41.	GÓDOR Balázs, KÁSA László Dr., SZÉKELY Bence	Híddaruk méretezési segédlete (2019.)
42.	FÜRJES Andor Tamás, KOTSCHY András, NAGY Attila Balázs, CSOTT Róbert	Teremakusztikai méretezés gyakran előforduló szituációkban
43.	KARÁCSONYI Zsolt Dr.	Faanyagok tartós szilárdsága Faanyagok szilárdságának változása az idő függvényében
44.	BALIKÓ Sándor Dr., ORBÁN Tibor, VARGA Péter, ZSEBIK Albin Dr.	Ötletlapok II. – Energiahatékonyság növelő ötletek egyszerű energetikai és gazdasági számításai
45.	PRIMUSZ Péter, PhD.	Hajlékony útpályaszerkezetek méretezése talajstabilizációk figyelembevételével
46.	NÉMETH Balázs, HÁMORI Sándor, KOSTYÁK Attila, VÍGH Gellért	Különböző funkciójú épületek klímatechnikája III. Segédlet ipari épületek lég- és klímatechnikai rendszereinek tervezése

47.	JANCSÓ Béla, KAVECZKI Gergely, KÓCZÁN Gábor, LABORCZI Tamás, KNOLMÁR Marcell, RAUM László	Csapadékvízgazdálkodás tervezési követelményei Hogyan tervezzünk városi csapadékelvezető rendszereket
48.	DOHÁNY Máté, SCHVANNER Norbert	Kerékpárosok sebességének felülvizsgálata jelzőlámpás csomópontokban
49.	JÓZSA Bálint, S. VÍGH Judit	Sebességcsökkentés hatásainak vizsgálata gyorsforgalmi utakon
50.	ZSEBIK Albin Dr., NOVÁK Dániel	Projektlapok I. – Energiahatékonyság növelő javaslatok projektlapjai
51.	MÓGA István Dr.	Beruházási projektek szabályozási és szabvány környezete, Tervezési követelmények meghatározása
52.	GÁBORI László Dr., BEINSCHRÓTH József Dr., NÓGRÁDI Gábor, RÁTKAY Tamás	Informatikai Tervező szakmai minősítő rendszere (Informatikai szakmai terület illesztése a Mérnök Kamarai működési rendbe és rendszerekbe) I. kötet: Koncepció és modell II. kötet: Modell illesztése III. kötet: Tudástár

53.	**VIRÁG Zoltán, GYURKOVICS Zoltán, SZAKÁL Szilárd, VIRÁG Zsolt, ORCSI Attila**	**Országos Tűzvédelmi Szabályzat épületgépész értelmezése a szakmai gyakorlatban** **Segédlet a gyakorló épületgépész mérnökök számára I.**

2020.

54.	KISS Jenő Dr., CSERMELY Gábor	JAVASLAT az egyszerű bejelentésű lakóépület megvalósításának – tervezés építés – módszerére
55.	SZILÁGYI Zsombor Dr.	A hidrogén a környezetbarát energiahordozó, Hidrogén az energetikában
56.	VARGA Tamás, SZEDENIK Norbert Dr., KOVÁCS Károly Dr., KRUPPA Attila, KULCSÁR Lajos, KAPITOR György, TURI Ádám	A nem norma szerinti villámvédelem egységes műszaki követelményrendszerének kialakítása és javaslat a teljes villámvédelmi szabályrendszer jövőbeli egységesítésére
57.	KÁDI Ottó	A gyalogosközlekedés közúti keresztezései
58.	MOLNÁR Szabolcs	„Hulladékból konnektorba” A települési szilárd hulladék energetikai hasznosításának lehetőségei
59.	VÁRDAI Attila	Segédlet szabadidős létesítmények tartószerkezeti tervezéséhez
60.	BEJÓ László Dr.	Szénlábnyom-elemzés készítése a faiparban

61.	JANCSÓ Béla, NÉMETH Gábor, SZIMANDEL Dezső	Szakmai útmutató vízilétesítmény tervezők számára a 2020 január 1-én hatályba lépett „VIZEK keretrendszer” használatához
62.	FELLEGI Zsóka, KARAFA Balázs, KOCH Edina, KOVÁCS Gábor, MURINKÓ Gergő, TÓTH Gergely József	Munkagödrök és földművek víztelenítése
63.	HOLÉCZY Ernő, OLÁH Róbert, SIKI Zoltán Dr., TAKÁCS Bence Dr., TÓTH Zoltán Dr., VARGA Tibor	Módszertani útmutató az elavult ingatlan-nyilvántartási térképek korszerű technológiákkal végzett felújításához
64.	GÁBORI László Dr., MOLNÁR Bálint Dr., NÓGRÁDI Gábor, RÁTKAY Tamás	Az Informatikai Tervező tervezési segédlete
65.	NÁDASDY Tamás, TOMASCHEK Tamás, PALÁSTY István, SZECSŐ Dániel Géza	Dinamikus forgalomirányítás tervezői segédlete gyorsforgalmi úthálózat esetén
66.	LENGYEL István	Szakmai útmutató szolgalmi jogok alapításához (mérnöki segédlet)

67.	**NÉMETH Balázs, SZLOVÁK Krisztián, VÍGH Gellért**	**Épületgépészeti tervezéshez praktikus, gyakorlati adatbázis**
68.	**FÜRJES Andor Tamás, BORSINÉ Arató Éva, NAGY Attila Balázs, ILLYÉS László, BORSI Gergely**	**Teremakusztikai méretezés gyakran előforduló szituációkban (példatár)**
69.	**BORBÁS Lajos Dr., GONDA Zoltán**	**Optikai feszültségvizsgálat – Kísérleti eljárás a konstrukció fejlesztésére, szerkezetek anyagfelhasználásának és teherviselésének optimalizálására**

2021.

70.	BLAZSOVSZKY László	A gázipar és a keményseprő-ipar határterületeinek szabályozási anomáliái a szakmagyakorlók és a felhasználók szemszögéből
71.	FORGÁCS Lajos Dr., NAGY Gábor, RÉV Zoltán	Kórháztervezés új szempontjai a 21. században - Korszerű kórházak infrastrukturális egységei
72.	HOLÉCZY Ernő, KISS Albert Miklós, KOVÁCS István, TAKÁCS Bence Géza Dr., TÓTH Zoltán Dr.	M.2.-2021. Mérnökgeodéziai tervezési segédlet
73.	BEJÓ László Dr.	Az ipar 4.0 alkalmazási lehetőségei a faipar területén
74.	BORBÉLY Dániel, HUDACSEK Péter, KARNER Balázs, KOVÁCS László, SÁNDOR Csaba	Monitoring, a geotechnikai kockázatkezelés eszköze
75.	FELFÖLDI Krisztina, JÁMBOR András, TÓTH Sándor, BÜKI Gábor, GÓDOR Balázs	Emelőgépek időszakos vizsgálatának eljárásrendje

76.	GYURKOVICS Zoltán, RÉBAY Lajos, NAGY Bernát	Szakmai útmutató az épületgépész felelős műszaki vezetők és műszaki ellenőrök számára
77.	ZSEBIK Albin Dr., NOVÁK Dániel, PAPP Ábrahám	Hulladékhő hasznosítás - hűtés és fűtés összekapcsolása Segédlet az elemzéshez és gyakorlati példák bemutatása
78.	CZINE Ferenc, HIRKÓ György	Elektromos meghajtású mikromobilitási eszközök - Jellemző paraméterek
79.	KALMÁR Tamás, LÁNYI Péter Dr., HÓZ Erzsébet	Kerékpárút hálózatok vizsgálata a fejlesztések és úthasználók tapasztalatai alapján
80.	VARGA Tamás, FARKAS Péter János, TOKODY Dániel Dr., ZSARNOVSZKI Attila, MÉSZÁROS Tamás, VERESS Árpád	Építményvillamossági tervezés robbanásveszélyes környezetben
81.	VONA Márton Dr., BALATONYI László Dr., TÉCSŐY István	Dombvidéki víz visszatartás, kisvízfolyások szabályozása természet közeli megoldásokkal Kisléptékű vízvisszatartás, kistelepülés-léptékű vízmegtartó megoldások

82.	**ZANATHY Valéria, BUZÁS Györgyi, TÓTH László**	**Acélszerkezetek korrózió elleni védelme – Acélszerkezetek korrózió elleni védelmére vonatkozó szabványok, előírások, szakami tapasztalatok összefoglalása**
83.	**JÓZSA Bálint, DOHÁNY Máté**	**DDI, avagy a fordított gyémánt csomópontok vizsgálata és magyarországi alkalmazhatósága**
84.	**SZÉPSZÓ Gabriella, ALLAGA-ZSEBEHÁZI Gabriella, LAKATOS Mónika, SZENTES Olivér, TAKSZ Lilla, SELMECZI János Pál, CZIRA Tamás Dr., CSÓKA Gergely, BAKA György**	**Éghajlatvédelmi vizsgálatok módszertana és az azt megalapozó adatbázisok alkalmazása**
85.	**ZSIGMONDI András, MARIÁN Gábor, WÉBER László**	**A műszaki egyenértékűség és helyettesítő termék egyenértékűségének megállapítási módjai**
86.	**NAGY János, HORVÁTH Rita, KAPITOR György, MERTLI Ferenc, PAPP Ábrahám,**	**Világítástechnika - segédlet az EKR dokumentáció készítéséhez – Alapismeretek és mintapéldák**

SITKU György,
ZSEBIK Albin Dr.

87. CSENDES János, VELLER Tamás

Épületautomatika – Összefüggésben az Energiahatékonysági Kötelezettségi Rendszerrel

2022.

88.	FÖLDI László József Dr., BERENCSI Bence	Ipari gépek CE jelölése és biztonsága az EU-s és hazai szabályozás tükrében
89.	SZILÁGYI Zsombor Dr., VADÁSZI Marianna Dr.	Irányelv új földgáz- és villamos energia szerződéskötéshez
90.	MÓCZÁR Balázs Dr., CSORBA Gábor, GRITSCH Ákos, KRISTON Gábor, MIHUCZ Tibor, SZENDEFY János Dr., SZILÁGYI Katalin	Segédlet ipari padlók geotechnikai és statikai tervezéséhez, kivitelezéséhez
91.	FELFÖLDI Krisztina, GÓDOR Balázs, NAGY Pál, RADVÁNYI G. Levente	G-D-36 Tanúsítvány kiadásához kompetencia-követelmények kidolgozása
92.	BUZÁS Zoltán, KÁLMÁN Miklós, BÖLSEI Tamás, LUKÁCS Tamás	A tervdokumentációk tartalmi és formai követelményeinek átdolgozása, különös tekintettel a Hír-Közmű bevezetésére. A Tervezés, Engedélyezés, Kivitelezés segédlet módosítása (92./1-2-3.)

93.	SIKI Zoltán Dr., CSEMNICZKY László, HOLÉCZYNÉ KAJTÁR Dóra, LEHOCZKY Máté, RÉPÁS Zoltán, TÓTH István	Szakmai útmutató digitális tervezési alaptérképek készítéséhez. A minőségi mérnöki munka segítése, a jó gyakorlat bemutatása, javaslat a térképek rétegszerkezetére és az alkalmazandó jelkulcsokra
94.	CSERMELY Gábor, TÓTH Péter	Szakmai útmutató a magasépítési kivitelezési munkák minőségellenőrzésére
95.	MARIÁN Gábor, ZSIGMONDI András	Az építési beruházások műszaki átadás-átvételi eljárása – Szakmai ajánlás az építési beruházások műszaki átadás-átvételi eljárására
96.	BARNA Sándor, MOLNÁR Tibor Dr.	Segédlet az AERMOD view szoftver használatához a légszennyező anyagok terjedési modellezéséhez
97.	BAKA György	A talajnak, mint természeti erőforrásnak a védelme a beruházások megvalósítása során
98.	BLAZSOVSZKY László	A gázipari szakmagyakorlók megváltozott felelőssége, hatásköre és a mindennapok gyakorlatának anomáliái a megváltozott jogszabályi környezetben

99.	FÜRJES Andor Tamás	Elektroakusztika elméleti és gyakorlati áttekintés
100.	RÁCZ Tibor, KUN Csaba, BALATONYI László Dr.	ITVT Integrált Települési Vízgazdálkodási Terv tervezési segédlet