Miskolci Egyetem

Gépészmérnöki és Informatikai Kar



Mérnökinformatikus szak

Termelésinformatika szakirány

Projekt ügyintézési feladat megoldása SAP rendszerben

Komplex feladat

Gazdik Tamás

EDJIVI

2018

**M I S K O L C I E G Y E T E M** Szám:GEIAK-……/20..

GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

Alkalmazott Informatikai Tanszék

**Komplex tervezési feladat**

**………………………………………………**mérnök informatikus jelölt részére

*A tervezés tárgyköre: Alkalmazásfejlesztés - Szoftvertechnológia*

*A feladat címe: Projekt ügyintézési feladat megoldása SAP rendszerben*

*A feladat részletezése:*

1. Ismerje meg a projektek felépítésének jellemzőit! Mutassa be a projektben elvégzendő feladatok jellemzőit és a végrehajtáshoz szükséges feltételeket!

Mutassa be a projektek modellezésének lehetséges megközelítéseit!

Ismerje meg és mutassa be az SAP rendszerek adattárolási megközelítését! Tervezzen és implementáljon SAP rendszerben egyszerű projektek ügyintézési feladatainak ellátására alkalmas megoldást!

Dokumentálja munkáját a komplex tervezési feladat keretein belül!

*Tervezésvezető:* Mihály Krisztián, egyetemi tanársegéd

*Konzulens:* Mihály Krisztián, egyetemi tanársegéd

Tartalomjegyzék

[1 A témaválasztás indoklása 4](#_Toc513755877)

[2 A komplex feladat ismertetése 4](#_Toc513755878)

[3 Elméleti áttekintés 5](#_Toc513755879)

[3.1 Projekt fogalmának ismertetése 5](#_Toc513755880)

[3.2 SAP rendszerek ismertetése 6](#_Toc513755881)

[3.3 ABAP 6](#_Toc513755882)

[4 Tervezés 7](#_Toc513755883)

[4.1 Modellezett szerepkörök 7](#_Toc513755884)

[4.1.1 Projekt felelős 7](#_Toc513755885)

[4.1.2 Projekt résztvevő 7](#_Toc513755886)

[4.1.3 Adminisztrátor 8](#_Toc513755887)

[4.2 Az alkalmazás funkciói 8](#_Toc513755888)

[4.2.1 Projekt felelős által elért funkciók 8](#_Toc513755889)

[4.2.2 Projekt résztvevő által elért funkciók 9](#_Toc513755890)

[4.2.3 Adminisztrátor által elért funkciók 10](#_Toc513755891)

[4.3 Alkalmazás komponensei, fő felépítés 11](#_Toc513755892)

[4.3.1 Maga szintű áttekintés. 11](#_Toc513755893)

[4.3.2 Felhasználói felület 11](#_Toc513755894)

[4.3.3 Projekt menedzsment API 12](#_Toc513755895)

[4.3.4 Adatmodell 13](#_Toc513755896)

[4.4 Tervezett felhasználói felület 15](#_Toc513755897)

[5 Megvalósítás 16](#_Toc513755898)

[5.1 Adatbázis modell megvalósítása SAP adatbázisban 16](#_Toc513755899)

[5.2 Üzleti logika rétege (funkciós modulok) 17](#_Toc513755900)

[5.3 Felhasználói felület (SAPGUI képernyők, navigációk, stb) 18](#_Toc513755901)

[6 Melléklet 19](#_Toc513755902)

[6.1 Ábrajegyzék 19](#_Toc513755903)

[6.2 Irodalomjegyzék 19](#_Toc513755904)

# A témaválasztás indoklása

Gyerekkorom óta az életemet folyamatosan végigkísérte a számítógépek világa. Kezdetben leginkább csupán a számítógépes játékok iránt mutattam érdeklődést, az idő múlásával azonban egyre inkább kíváncsivá tett, hogy hogyan is működnek az egyes számítógépek, miért működnek úgy, ahogy teszik azt, illetve a különböző lehetőségek, amik elérhetőek számunkra a cégek által kiadott szoftvereken kívül. Ekkor kezdtem el ismerkedni a programozással, elsőként pedig a HTML és C nyelvvel.

Gimnáziumi tanulmányaim során, az informatika órán szerzett tudásanyagot meglehetősen szegényesnek tartottam, ezért döntöttem úgy, hogy a középiskola befejeztével informatikai tanulmányokat folytatok, ahol részletesebben megismerkedhetem a számítógépek világával.

Úgy tűnt, hogy a mérnökinformatikus képzés helyes döntésnek bizonyult. Hamarosan elérkezett a szakirány választás ideje is. Az induló szakirányok közül a termelésinformatika nyerte el tetszésemet. A szakirány tárgyainak keretein belül ismerkedhettünk meg többek között az ERP rendszerekkel és köztük az ezen a területen iparágvezető vállalattal, az SAP-val. Látva, hogy az elkövetkezendő időkben is milyen jelentős szerep jut majd az integrált vállalatirányítási rendszereknek, úgy döntöttem, hogy a jövőben ezzel a tématerülettel és ezekkel a rendszerekkel fogok foglalkozni. Az SAP előadásokat látogatva pedig világosan körvonalazódott előttem, hogy a komplex tervezési feladatomat és a szakdolgozatomat is ebben a témában szeretném kidolgozni, egyebek mellett az ABAP nyelvet használva.

# A komplex feladat ismertetése

A komplex tervezési feladatként SAP rendszerben, ABAP fejlesztési nyelven hozok létre projektmenedzsment feladatok támogatására való/szolgáló megoldást. Elsőként ismertetem a projekt fogalmát, feltérképezem a szükséges funkciókat és komponenseket az alkalmazással kapcsolatba lépő szereplőkkel együtt és modellezem a projektet. Végül SAP rendszerben, ABAP fejlesztési nyelven hozok létre projektmenedzsment feladatok támogatására való megoldást SAP NetWeaver alapokon.

# Elméleti áttekintés

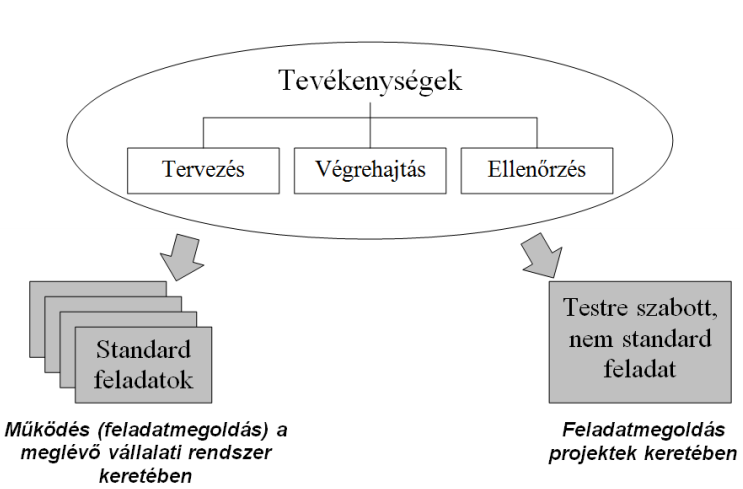
## Projekt fogalmának ismertetése

*„A projekt konkrét, általában egy meghatározott  
költségvetési és időkereten belül elérendő cél  
megvalósítására ideiglenes jelleggel összeválogatott  
emberek és egyéb erőforrások csoportja.”*

(Robert J. Graham)

Napjainkban a vállalatok működése során előállított termékek és szolgáltatások egy alaposan megtervezett és megfelelően kivitelezett folyamatos és ismétlődő termelési és gyártási folyamat részeként jönnek le.

A modern vállalatokban azonban beérkezhet olyan megrendelés is, amikor a terméket vagy szolgáltatást egyéni igényeknek megfelelően kell előállítani, ebben az esetben nem sorozatgyártott gyártmányról van szó. Az ilyen egyszeri alkalomból megtervezett, konkrét célok elérésére irányuló tevékenységsorozatot nevezzük projektnek. [1]



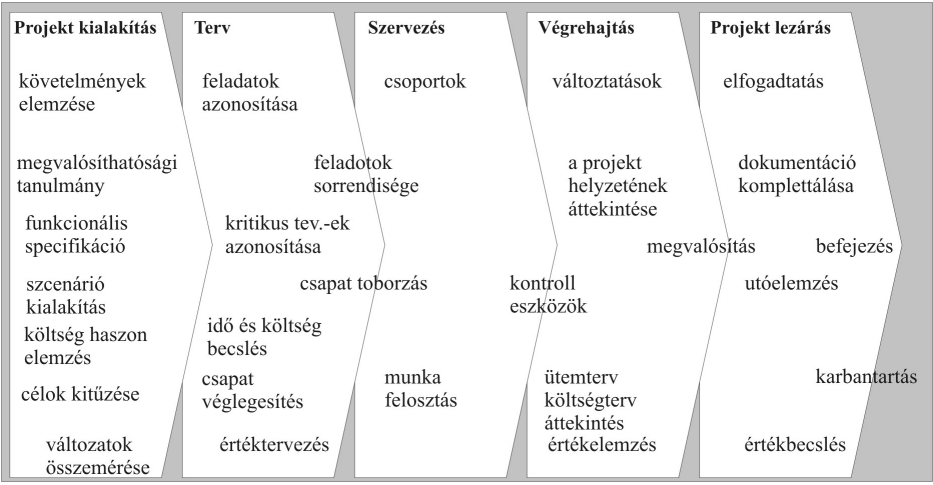
1. ábra: A vállalaton belül szereplő főbb tevékenységtípusok

A projekt elnevezés a latin *projectum* szóból ered, melynek szó szerinti jelentése: „előre vetítés”. Jelentéséből kiindulva a projekt inkább jelenti magát a tervezést, mintsem a teljes kivitelezést. Ahhoz, hogy ténylegesen megértsük a projektek lényegét, érdemes megvizsgálni, hogy milyen tulajdonságokkal rendelkeznek:

* minden egyes projekt egyedinek tekinthető, többek között abban az értelemben, hogy sajátos kezdési és befejezési időpontjai vannak, ugyanakkor ezek az időben is egyediek, tehát egy adott projekt egy adott időintervallumra korlátozódik le, csak egyszer megy végbe
  + legtöbb esetben egyéni igények kielégítésének céljából jönnek létre, így végeredményük (amely legtöbb esetben termék vagy szolgáltatás) különbözik a sorozatgyártásban előállított termékektől vagy szolgáltatásoktól. Ez a különbség jelentkezhet például az előállítás módjában vagy konstrukcióban
  + mivel egyedi termék vagy szolgáltatás céljából jön létre, így a kockázat is nagyobb, hiszen az eddig jól bevált módszerektől némileg eltér a folyamat
  + saját, hozzárendelt erőforrásokkal rendelkezik, ezek lehetnek pénzügyi, eszköz-, humán-, és anyag erőforrások

A fentiekből tehát láthatjuk, hogy a projektekre általánosan jellemző, hogy olyan, egymással összefüggő tevékenységsorozatot jelentenek, amelyek konkrét célok és eredmények elérése érdekében, minimális erőforrás felhasználásra törekedve, vállalható kockázat mellett, bizonyos minőségi és mennyiségi követelményeknek megfelelően, egyértelműen meghatározott terméket vagy szolgáltatást eredményeznek.

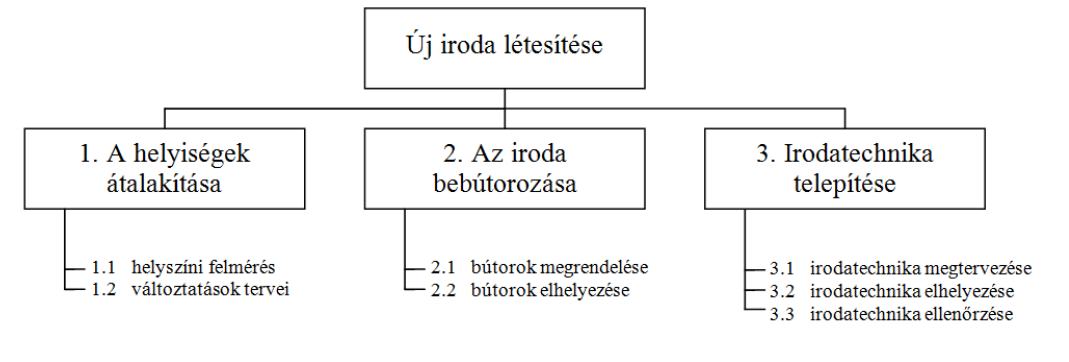
A projektek kezdési és befejezési időpontjaik, valamint végrehajtásukhoz szükséges időszeleteket figyelembe véve elmondható, hogy a projektek az élőlényekkel analóg módon életciklussal rendelkeznek: „megszületnek” vagy létrejönnek, „tartanak és kiteljesednek”, majd „elhalnak”, tehát befejeződnek vagy lezárulnak. Ezt az életciklust remekül szemlélteti a Weiss és Wysocki által megalkotott ötfázisú modell (1994):



2. ábra: A Weiss és Wysocki féle ötfázisú projekt-életciklus modell

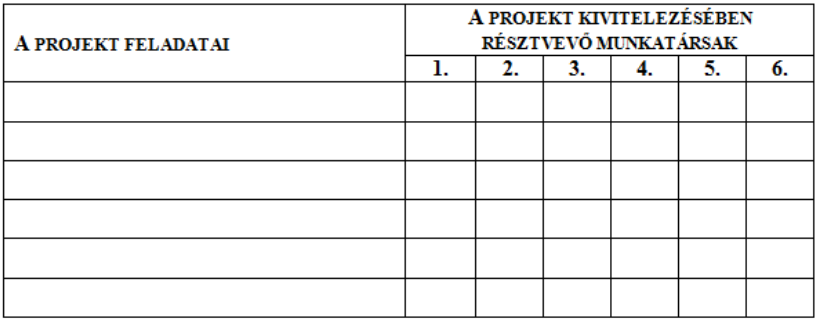
A modell alapján a legelső szakasz a projekt kialakításának szakasza. Ennek kezdetét a projekt megnevezése, a projekt céljának és a projektmenedzser rögzítése jelenti a projektalapító okiratban. Ebben a szakaszban projektjavaslatokat dolgoznak ki, amelyek majd elindítják magát a projektet, amennyiben megfelelőre értékelték és jóváhagyták a megvalósítást. A szakasz végén elkészül a munkakimutatás, amely tartalmazza többek között a célokat a hozzájuk tartozó mérhető mutatókkal és indikátorokkal, az elérendő eredményeket, a projekt hatókörét, illetve a költség-, és ütemtervbecslést.

Pozitív elbírálás esetén a projekt a második, tervezési fázisba léphet. A tervezés első lényeges lépése meghatározni a projekt sikeres végrehajtásához vezető feladatokat. Ezt legegyszerűbben a WBS (Work Breakdown Structure vagy munkalebontási szerkezet) segítségével tehetjük meg. A WBS lényegében egy faszerű struktúra, amelynek gyökere a projekt célja, végeredménye. Ezt bontjuk fel különböző munkacsoportokra, az egyes munkacsoportokat pedig kisebb csoportokra, végül ezeket tovább bontva jutunk el a konkrét, végrehajtandó, külön-külön kezelhető feladatokhoz, amelyeket szakemberek fognak elvégezni.



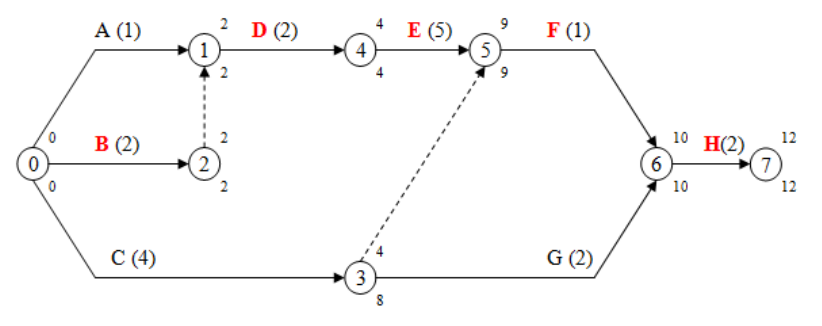
3. ábra: Új iroda létesítésének egy lehetséges WBS vázolása

A feladatok meghatározását követően megfelelő kompetenciával rendelkező felelősöket rendelhetünk azokhoz. Ennek legegyszerűbb szemléltetése kompetencia és felelősségi mátrix segítségével történik. A mátrix minden egyes sora egy-egy feladatot jelöl, az oszlopokban pedig a projektben résztvevők találhatók. Felelősségi-, illetve feladatköröket kialakítva egy adott feladat bizonyos személyeket tekintve különböző mezőértékeket kaphat. (pl. D – dönt a végrehajtásról, V – végrehajt, É – értesít, vagy visszajelzést küld).



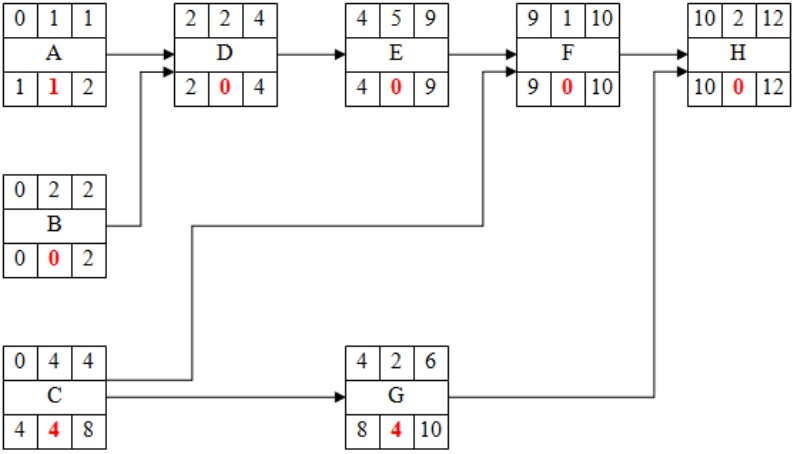
4. ábra: Felelősségi és kompetencia mátrix lehetséges felépítése

Miután meghatároztuk, hogy MIT és KI fog végrehajtani, a következő fontos feladatunk a HOGYAN megválaszolása lesz. Ez lényegében a logikailag kötődő feladatok végrehajtási technológiájának megtervezését jelenti és eredménye a projekt hálóterve. A legismertebb tervezési technikák a CPM, MPM és PERT.  
A **CPM** (Critical Path Method) lényegében egy matematikai algoritmus, amely az adott projektben fellépő feladatokat, azok végrehajtási időit és a köztük lévő függőségeket (pl. megelőzési relációk) felhasználva határozza meg a leghosszabb végrehajtási útvonalat (ami egyben a kritikus aktivitásokat tartalmazó kritikus út lesz), illetve az egyes feladatok legkorábbi és legkésőbbi indítási és befejezési időpontjait oly módon, hogy a projekt végrehajtási ideje ne változzon. A meghatározott kritikus utat, illetve a projekthez szükséges végrehajtási időt az aktivitások párhuzamosításával, illetve további erőforrások hozzárendelésével javíthatjuk. Az alábbi ábrán egy ilyen CPM módszerrel meghatározott hálótervet láthatunk. A csúcspontokban számokkal az eseményeket, jobb felső és alsó részen a leghamarabbi kezdési, illetve befejezési időket találjuk. Az irányított élek nagybetűvel a tevékenységeket, zárójelben a végrehajtáshoz szükséges időegységek számát reprezentálják.



5. ábra: Példa CPM-el meghatározott hálótervre

Az **MPM** technika a CPM-el ellentétben csupán tevékenységeket tartalmaz, amelyek ábrázoláskor a csomópontokba kerülnek. Események helyett mérföldköveket (pl. fizetési pontok, információáramlás egyik projektszereplőtől a másiknak, előrehaladási jelentés) lehet ábrázolni speciális, nulla időtartamú tevékenységekként. A lenti ábrán a négyzetek jelölik a tevékenységeket. A bal oldali számok a legkorábbi kezdési és befejezési, a jobb oldaliak a legkésőbbi kezdési és befejezési időpontokat jelölik. Középen fent a végrehajtási idő, lent a tartalékidő található. Az irányított nyilak megelőzési, illetve követési relációt jelölnek, ennek 4 típusa van: BK (befejezés – kezdés), KK (kezdés – kezdés), BB (befejezés – befejezés), és KB (kezdés – befejezés). Az ábrán csupán BK kapcsolat található késleltetési idő (*lag time*) nélkül.



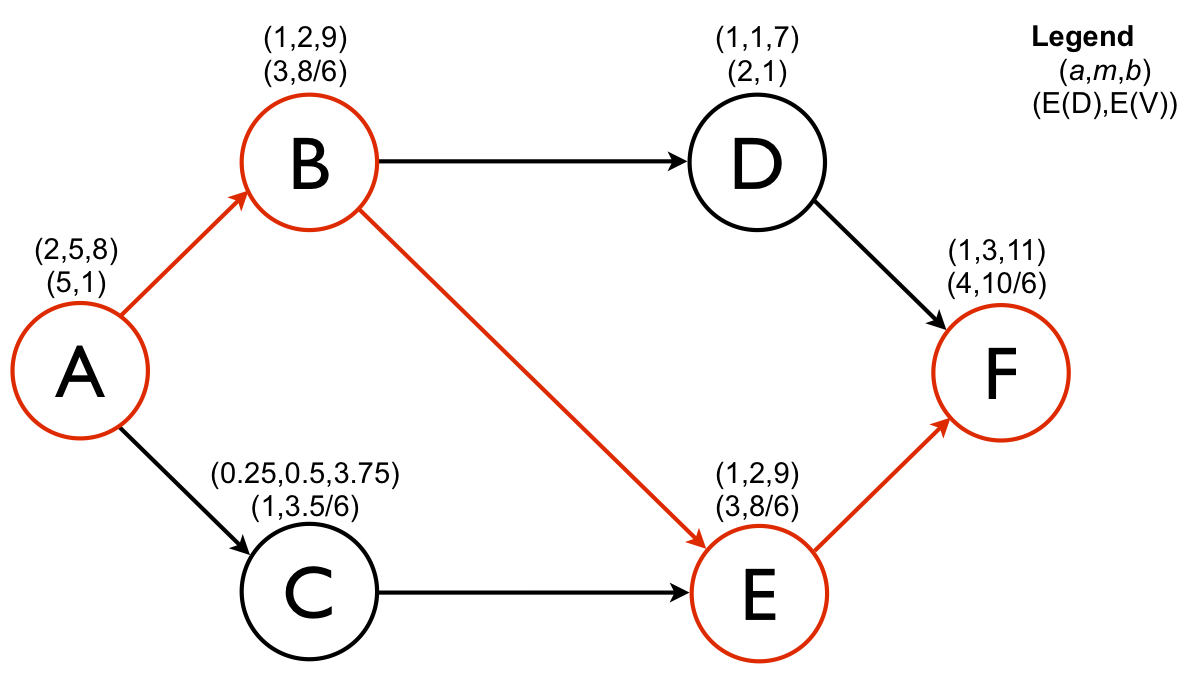
6. ábra: Példa MPM-el készített hálótervre

A **PERT** módszer (Program (vagy Process) Evaluation and Review Technique) segítségével egy adott cél eléréséhez szükséges feladatok vagy tevékenységek időbeli végrehajtását, a köztük fennálló logikai, belső kapcsolatokat, illetve függőségeket tudjuk ábrázolni. A modellezés végeredménye szintén egy hálóterv lesz, amelyet az előzetesen már ismert események, tevékenységek és a végrehajtáshoz szükséges idők segítségével építünk fel. Az események tulajdonképpen tevékenységek kezdetét vagy befejezését jelölik és nincs időbeli vagy másféle erőforrás szükségletük. Az ábrázolás során körökben jelennek meg. A tevékenységek az eseményeket kötik össze és irányított nyíllal jelölik őket. Az eseményekkel szemben végrehajtásuk időigényes, valamint erőforrást is használhatnak (humán, eszköz, hely, stb.). A végrehajtási idők meghatározásánál a PERT három különböző végrehajtási időt felhasználva tervez. Ezek az

* optimista végrehajtási idő: a lehető leggyorsabb időszelet, ami alatt végrehajtható a tevékenység
* a pesszimista végrehajtási idő: a végrehajtáshoz szükséges leghosszabb időtartam
* valószínű végrehajtási idő: a végrehajtáshoz szükséges erőforrások és egyéb állapotokat figyelembe véve, normális esetben mennyi idő alatt hajtható végre a tevékenység

A fentieket súlyozottan figyelembe véve határozható meg egy ún. várt vagy becsült végrehajtási idő, amelynek kiszámítása a

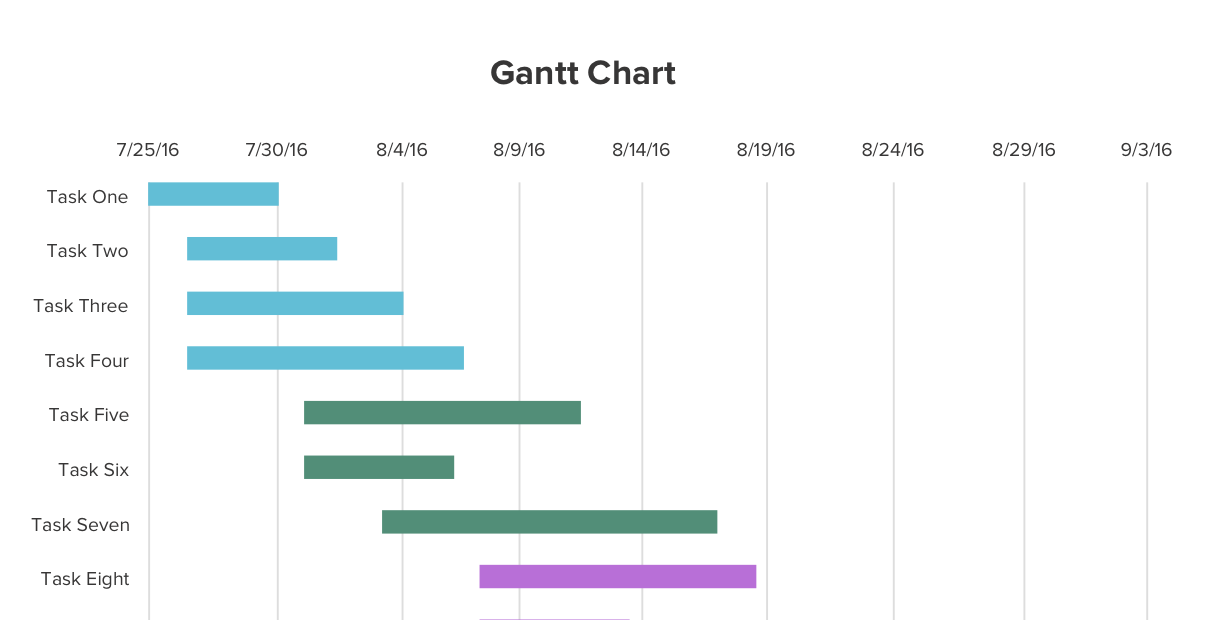
képlet segítségével történik.



y. ábra: PERT módszer segítségével készített hálóterv

Az y. ábrán látható hálóterven az egyes események jelölése az ABC nagy betűivel történik. A zárójelben szereplő számok rendre az optimista, a valószínű, valamint a pesszimista végrehajtási időket adják meg. Az ebből a 3 értékből a fenti képlet segítségével számolt várt végrehajtási időt jelöljük az ábrán E(D)-vel (expected duration). A variancia jelölése pedig E(V)-vel (expected variance) történik, értékének kiszámításához pedig a vennünk kell a pesszimista és optimista végrehajtási idők különbségét, az értéket pedig osztanunk kell 6-al.

Ugyan az elkészült hálóterv nem szükséges feltétele a Gantt-diagram készítésnek, azonban mivel a tevékenységek közötti logikai sorrend és a végrehajtási idők már ismertek, így felhasználva azt gyorsabb a diagram elkészítése. A Gantt-diagram nevét Henry Laurence Ganttról kapta, aki 1910 és 1915 között dolgozta ki kezdeti változatát. Ez szemben a továbbfejlesztett Gantt-diagrammokkal, nem adott módot tevékenységek közötti függőségek vagy a nem kritikus úton szereplő tevékenységekhez tartozó időtartalékok ábrázolására sem. A sávos ütemterv néven is ismertté vált diagramot a század közepéig önálló tervezőeszközként használták, majd a hálós tervezési módszerek népszerűvé válása után ezeket együttesen alkalmazva váltak a mai napig az egyik leginkább célravezető projekttervezési rendszer részévé.   
A Gantt-diagramon vízszintes irányban található az időtengely, függőlegesen pedig jellegzetesen az ábrázolandó tevékenységek találhatók. Ezekkel egy vonalban történik a végrehajtás szemléltetése tipikusan egy vízszintes vonallal, amelynek kezdőpontja a tevékenység indításához tartozó időpillanat, végpontja pedig a befejezés időpillanatának felel meg.

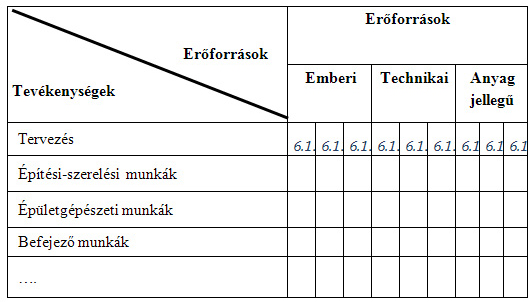


. ábra: példa Gantt-diagramra

Mint azt már fentebb említettem, ahhoz hogy az adott projektet felépítő feladatokat végre tudjuk hajtani, különböző típusú erőforrásokra lesz szükségünk. A projekt teljesítéséhez szükséges összes erőforrásnak szerepelnie kell az erőforrástervben, ami ezen tervezési fázis végeredményét adja. A csoportosítás a szakirodalom nagy részében egységes, az alábbi erőforrástípusokat felhasználva tudunk feladatokat végrehajtani:

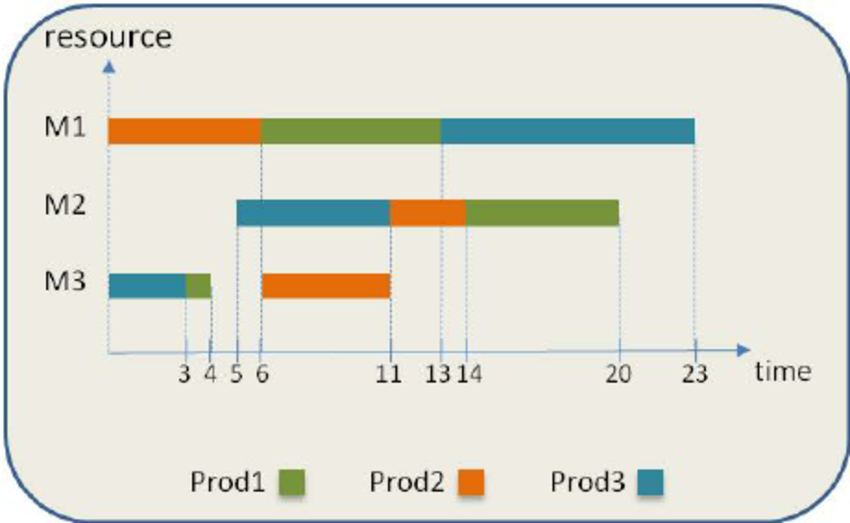
* munka típusú erőforrások: ide tartoznak az emberi, azaz humán, illetve a technikai erőforrások (gépek, berendezések, a különböző eszközök)
* anyag típusú erőforrások: a projekteredményhez felhasznált vagy az abba beépülő eszközök és egyéb elemek
* költség típusú vagy pénzügyi erőforrás: olyan jellegű szolgáltatások költségeinek elszámolására használják, amelyek nem idő vagy mennyiség alapon történik (pl. egy beszállító által kiszámlázott szolgáltatás ellenértéke)

Az egyes tevékenységekhez szükséges erőforrások mennyiségének tárolása egy erre a célra létrehozott mátrix segítségével történhet, az alábbi kép szerint



2. ábra: példa tevékenység-erőforrás-szükséglet mátrixra

Az egyes erőforrástípusok időbeli igénybe vétele Gantt-diagram segítségével ábrázolható, ahol függőleges tengelyen szerepel az adott erőforrástípusból szükséges mennyiség tevékenységenként lebontva, vízszintesen pedig az időtengely helyezkedik el. Egy másik megközelítésben az egyes erőforrás-előfordulások igénybevétele szemléltethető hasonló módon.



3. ábra: az egyes erőforrások (M1, M2, M3) igénybevétele különböző folyamatok (Prod1, Prod2, Prod3) végrehajtásához az idő függvényében

A tervezési fázis egyik sarokpontját adja a költségtervezés, amely során feltérképezzük és megtervezzük a projekt kivitelezéséhez szükséges költségeket az előzőekben meghatározott szabályokat betartva, piaci viszonyoknak reálisan. A költség tervezésére az egyik legelterjedtebb és legpontosabb módszer a tevékenységalapú költségtervezés. Ezt a tevékenység hálótervek, valamint az időszükségletek és erőforrás-szükségletek kidolgozását követően tudjuk alkalmazni. Két fő komponense a tevékenységekhez rendelt fix költségek, illetve az erőforrásokhoz rendelt költségek. Előbbi esetében a tevékenység időtartamától, továbbá az elvégzett munkától független, az utóbbi esetében fajlagos összegekről (munkadíj esetén pénznem/óra, anyag költség esetén pénznem/m2) vagy költségekről beszélhetünk.

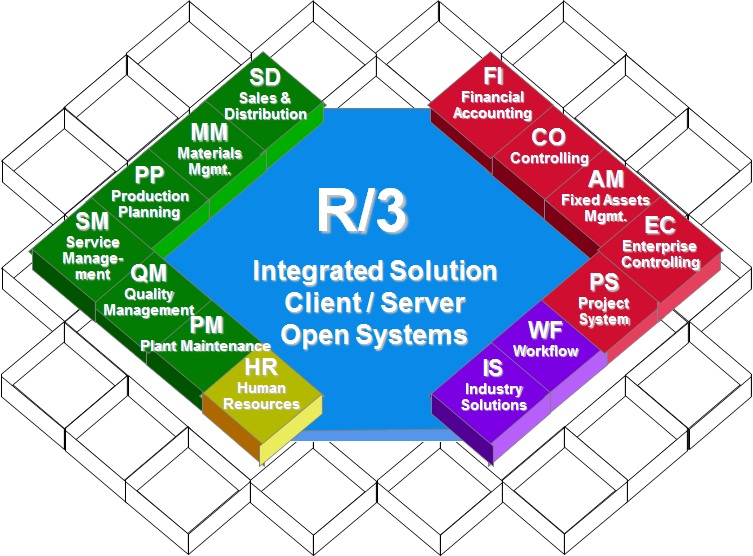
## C:\Users\sparch\AppData\Local\Microsoft\Windows\INetCache\Content.Word\What_is_ERP.JPGSAP rendszerek ismertetése

Ahhoz, hogy betekintést nyerhessünk abba, hogy tulajdonképpen mi is az az SAP, először érdemes megismerkednünk az integrált vállalatirányítási rendszerekkel vagy idegen szóval ERP rendszerekkel (ERP = Enterprise Resource Planning). Ezek lényegében olyan egységes információrendszerek, amelyek a vállalat egészére nézve, annak valamennyi folyamatát bevonva valósítanak meg integrációt. Egy másik megfogalmazás szerint integrált vállalatirányítási rendszer alatt egy adott vállalat legtöbb folyamatát lefedő szoftvercsomagot értünk. Manapság a legtöbb integrált vállalatirányítási rendszer azonban nem csak a folyamatos működéshez szükséges pénzügyi, humán és technikai erőforrások kezeléséhez, tervezéséhez és nyomon követéséhez kötődő feladatokban nyújt segítséget, hanem a vállalat határain túlnyúló folyamatok, mint az ügyfélkapcsolat-menedzsment (CRM = Customer Relationship Management), szállítási lánc menedzsment (SCM = Supply Chain Management), stb. végrehajtását is segíti. Ezekből láthatjuk, hogy napjainkban nem individuális vállalatokról, hanem sokkal inkább vállalatok hálózatáról vagy virtuális vállalatokról érdemes beszélnünk. Virtuális vállalat alatt pedig olyan önálló vállalatok vagy szervezeti egységek ideiglenes vagy meghatározott időpontig terjedő, közös céljaik megvalósítására szolgáló együttműködéséről beszélünk, amelyek területileg nem feltétlenül egy helyen vagy egy régióban találhatók.

Az SAP AG (eredetileg a SystemAnalyse und Programmentwicklung német kifejezésből ered, később a Systeme, Anwendungen und Prodekte, magyarul Rendszerek, alkalmazások és termékek elnevezés vált elterjedté) napjainkban világelső integrált vállalatirányítási rendszer. 1972-ben alapította öt, az IBM-ből kilépő rendszerelemző szakember a németországi Walldorfban. Kezdetben az volt a célkitűzésük, hogy megkönnyítsék a vállalatok egyes folyamatait a megrendeléstől kezdve egészen a számlázásig, illetve, hogy különböző megoldásokat találjanak ki a versenyképesség fenntartásához szükséges irányítás támogatására.

Első kiadott verziója SAP R/2 (R = real, valós idejű; a rendszer adott feladatot minden egyes esetben ugyanannyi idő alatt hajt végre, ez az időtartam nem számít) néven futott és 1979-ben jelent meg. A nyolcvanas évek számítástechnikai lehetőségei miatt főleg csak nagyvállalatok *mainframe* gépein, vagyis nagy számítóközpontokban használták, emiatt a kis- és középvállalkozások nem részesülhettek a használat által nyújtott előnyökből. A kilencvenes évek elején azonban megjelentek és egyre nagyobb számban terjedtek a nagyobb számítási kapacitású és egyre alacsonyabb vételi áron kínált számítógépek, aminek köszönhetően más rétegek számára is elérhetővé vált az SAP.

Végül ez a változás vezetett el 1992-ben az SAP legismertebb verziójához, R/3-hoz, amely kliens-szerver architektúrával, egységes felhasználói felülettel, dedikált adatbázis használattal és stabil szerver támogatottsággal rendelkezik. A rendszer továbbá moduláris felépítésű, azaz a vállalatok különböző folyamatait különböző modulokon keresztül támogatja.



4. ábra: Az SAP R/3 moduljai: zöld színnel a logisztikai, sárgával az emberi erőforrás, pirossal a pénzügyi, lilával pedig az egyéb kiegészítő modulok

A logisztikai modulok közül a legelterjedtebbek és a hozzájuk tartozó funkciók a teljesség igénye nélkül az

* SD (Sales and Distribution): vevői megrendelések rögzítése, nyomon követése, értékesítési törzsadatok kezelése
* MM (Materials Management): anyagbeszerzés, rendelés, anyagkezelés, készletezés, leltározás, raktározás tervezés, könyvelések
* PP (Production Planning): termeléstervezés, különböző termelési folyamatok támogatása
* QM (Quality Management): minőségtervezés, felülvizsgálat, minőségügyi tanusítványok kezelése

A pénzügyi modulok közül a legelterjedtebbek az

* FI (Financial Accounting): pénzügyi és számviteli információk kezelése, könyvelések menedzselése, valós idejű bevétel- és kiadás-kimutatás
* CO (Controlling): gazdasági és stratégiai döntéshozatal támogatása, költségirányítás, költséghely- és árbevétel-számítás segítése
* TR (Treasury): számlakimutatások, pénzügyi menedzsment, piaci kockázatok elemzése, kezelése
* IM (Investment Management): költségvetés kezelése, tervezése, beruházásokhoz kapcsolódó funkciók segítése

Emberi erőforrás menedzsment modul

* HR (Human Resources): személyügy, felvétel, toborzás, bérszámfejtés, utaztatás és a vele járó költségek, személyiségfejlesztés, tréningek, rendezvényszervezés, stb.

A klasszikus SAP R/3 egyik legnagyobb erőssége, hogy hardware független. Ezt úgy valósítják meg, hogy a forráskód az első hozzáférés során az adott operációs rendszerre fordul, majd a bájtkód eltárolódik az adatbázisban. Hagyományos esetben ez az operációs rendszer által kezelt fájlrendszerben raktározódna el. Érdemes megemlíteni, hogy a rendszer alapja szintén adatbázis független, illetve az egyetlen megkötés, hogy olyan racionális adatbázis legyen, amelyben a különböző utasítások és kifejezések megfelelnek az OSQL (Open SQL) szabványnak, így sajátos, arra az adott adatbázisra specifikus kifejezéseket nem tartalmazhatnak.

A kétezres évek elején megjelenő a felhő alapú, mobil és memória alapú számítások pedig új távlatokat nyitottak a valós idejű adateléréshez. Az SAP a 2010-es évek elején, mint legsikeresebb ERP rendszer jelentette meg a piacon az újabb generációt, az SAP S/4HANA-t, amely memória alapú adatbázison alapul, aminek köszönhetően rendkívül gyors és pontos adatfeldolgozásra és –elemzésre képes. Az előzőekben ismertetett tulajdonságain túlmutatóan pedig támogatja a korszerű felhasználói felületi technológiákat is (ilyenek a JavaScript, UI5, stb.).

## Az SAP NetWeaver és az ABAP nyelv

Az SAP NetWeaver az SAP technológiai platformja, amely olyan komponensek gyűjteményét szolgáltatja a vállalatok felé, amely segít bizonyos problémák leküzdésében vagy SAP környezetbeli feladatok megoldásában. Leginkább alkalmazások fejlesztésére, felhasználók összekapcsolására, folyamatok összehangolására, rendszerek kezelésére, adatok és információk kezelésére vagy biztonsági megoldások esetén használják.

Rendelkezik egy applikációs szerverrel, amin keresztül futási környezetet biztosít az egyes SAP alkalmazás számára. Ilyenek többek között a Business Suite vagy Üzleti Megoldások csomagba tartozó SRM (*supplier relationship management* vagy beszállítói kapcsolati menedzsment), CRM (*customer relationship management* vagy vevői kapcsolat menedzsment), SCM (*supply chain management* vagy az ellátási lánc menedzsment), PLM (*product lifecycle management* vagy termék életciklus menedzsment), végül a TMS (*transportation management system* vagy ???).

A NetWeaver része egy eseményvezérelt központi folyamatütemező (Central Scheduler) is, amely központilag automatizálja és felügyeli a háttérben futó és az üzleti alkalmazások végrehajtását.

Főbb feladati közé tartozik az emberek (közvetlenebb környezet és könnyebb együttműködés kialakítása az egyes felhasználók között), az információ (a különböző forrásokból és formátumokból érkező adatok és információk feldolgozásának segítése azáltal, hogy egységességet teremt), és folyamatok (az egyes vállalatok, ágazatok közötti munka zökkenőmentes folyásának elősegítése. Ezáltal a különböző platformokon futó szolgáltatások, alkalmazások és egyéb folyamatok akadálymentesen kommunikálhatnak) egy homogén SAP környezetben történő integrálása.

Mivel a NetWeaver nyílt architektúrára épül, így maga is főleg nyílt standardú technológiákat (főleg XML és Java környezetben ismerteket) használ. Főként az ABAP nyelvet használva fejlesztik, de akadnak benne C, C++, és Java elemek egyaránt.

A fent említett funkciókon és szolgáltatásokon felül az SAP NetWeaver rendelkezik különböző funkcionalitásokat reprezentáló komponensekkel is, ezek

* SAP Business Intelligence: különböző elemzéshez és reportáláshoz (?) szükséges eszközöket biztosít
* SAP Enterprise Portal: egyszerű, egylépcsős belépési pontot biztosít a különböző alkalmazások felé, feltéve, hogy az illető rendelkezik a szükséges jogosultságokkal
* SAP Exchange Infrastructure: a vállalat belső SAP környezete és külső rendszerek közötti adatcserét segíti
* SAP Master Data Management: segítségével a vállalatok a legfontosabb törzsadatokat heterogén forrásokból központi tárhelyekbe juttathatják át és tárolhatják, elemezhetik
* SAP Knowledge Warehouse: iparág-specifikus ismeretek, útmutatók és tananyagok eltárolásának támogatása egy tudásbázisban későbbi használatra

Az SAP NetWeaver Applikációs Szervere futási környezet biztosításával lehetőséget nyújt arra is, hogy a nyílt kódú programokat tovább bővíthessük ABAP programozási nyelven.

Az **ABAP** (Advanced Business Application Programming) az SAP által kifejlesztett magasszintű, 4GL, típusos, procedurális, interpreteres programozási nyelv, amelynek segítségével SAP környezetben hozhatunk létre különböző üzleti alkalmazásokat. Az SAP forráskódja vevőik számára nyíltan elérhető, azt bármikor módosíthatják vagy bővíthetik az ABAP nyelv segítségével.

Az SAP R/1 és R/2 idején riportok megírására szolgált, ekkor az SAP rendszert főleg C nyelven fejlesztették. Az SAP R/3 megjelenését követően az ABAP nyelvet is fejlesztették és így jött létre az ABAP/4, utalva 4. generációs programozási nyelv létére. Ettől kezdve a rendszermagon kívül minden alkalmazást és fejlesztés ABAP nyelven írtak meg. A '90-es évek végére megjelent az ABAP Object, amely az objektumorientált paradigmáknak megfelelő továbbfejlesztése volt a nyelvnek.

A fejlesztés és a programok futása egyaránt az applikációs szerveren fut, így éri el a platformfüggetlenséget a nyelv.A nyelvben integráltan jelennek meg az adatbázis kezelő műveletek az OpenSQL-nek megfelelő módon.

Az ABAP nyelvet főként az SAP fejleszti. Programkód írására a nyílt kódú Java fejlesztői környezet, az Eclipse-ben használt pluginok vagy az SAP rendszerbeli Workbench (Wikipédia - HU + EN!!! https://www.tutorialspoint.com/sap/sap\_programming\_language.htm) tranzakció segítségével van lehetőségünk. Ezen módszerek egészen a kód megírásától az automatizált tesztelésig nyújtanak szükséges eszközöket.

# Tervezés

## Modellezett szerepkörök

Az irodalomkutatás elvégzése után az alábbi üzleti szerepeket határoztam meg:

* üzleti elemzők (analyst vagy business analyst): felelőssége meggyőződni arról, hogy a vevői igények megfelelően rögzítettek és dokumentáltak, illetve hogy a projekt a vállalat szempontjából hozzáadott értékkel rendelkezik.
* vevő (client): az a szereplő, akinek a kérésére a projekt létrejön, a projektet megrendelő
* vevő oldali projekt menedzser, projektfelelős (client project manager): nagyobb projektek esetén szükség lehet valakire, aki vevő oldalon felelős a projektért, ő lesz az elsődleges kontakt is.
* adatbázis adminisztrátor (database administrator): a szükséges adatbázisokat és táblákat modellező, tervező és létrehozó személy.
* tervező (designer): az üzleti igényeket kielégítő megoldás megtalálásáért és megtervezéséért felelős. Rendelkeznie kell széles körű technológiai tudással, amelyet felhasználva a projekt végrehajtása során optimum hatást érhet el.
* szakember, projekt résztvevő (expert): teljes- vagy részidős humán erőforrás, aki felelős az elvégzendő munka megértéséért, elvégzéséért, dokumentálásáért és a visszajelzésért. Származhatnak egy vagy több funkcionális szervezetből is. Együttesen adják a projekt csapatot (project team).
* projekt menedzser, projektfelelős (project manager): a projekt irányításáért felelős személy. Ide tartozik a cél meghatározása, a projekt folyamatos, célirányos vezetése, szükséges lépések, erőforrások meghatározása, a költségvetés és aktivitások ütemezése, folyamatos felügyelet, mérföldkövek kijelölése, kockázatok kezelése, konfliktusok feloldása.
* támogató, szponzor (sponsor): legmagasabb szinten a projektet a végrehajtásért felelős szponzor támogatja, azaz pénzügyileg finanszírozza a projektet, hatáskori változtatásokat hagy jóvá, biztosítja a szükséges erőforrásokat, elfogadja vagy elutasítja az eredményeket. Ugyanakkor felelősségi körébe tartozik a magas szintű problémamegoldás és irányítás.
* stakeholder: a projekt kimenetében érdekelt szereplő. Belső stakeholder lehetnek belső vevők, menedzsment, adminisztrátorok, stb. Külső stakeholderek közé soroljuk a beszállítókat, befektetőket, a különböző kormányzati szervezeteket. Magasabb szintű stakeholderek a szponzorokkal együttesen határozzák meg a stratégiai irányzatot, biztosítják az erőforrásokat, a költségvetést, az ütemtervet és alkalmazzák a szükséges változtatásokat.
* csapatvezető (team leader): nagyobb projekt esetén van rá szükség. Információt gyűjt az alsóbb szintekről, ezek felhasználásával a kitűzött cél eléréséhez szükséges aktivitások felé tereli a figyelmet, nem utasít. Megfelelő kompetenciával rendelkezve nem csupán irányít, aktívan részt is vesz a munkában, az elvégzendő feladatokban. Az alá beosztott projekt résztvevők által végzett aktivitások eredményeit, a felmerülő kockázatokat, további erőforrás igényeket pedig közvetlen kapcsolatot tartva jelenti a projekt menedzsernek.

A fent bemutatott fontosabb felmerülő szerepkörök közül jelen esetben a *rendszer adminisztrátor*, mint adatbázis adminisztrátor és tervező egy személyben, a *projektben résztvevő* vagy szakember, a projekt menedzser vagy *projektfelelős*, illetve opcionálisan a csapatvezető vagy *team leader* szerepeket kívánom megjeleníteni a tervezés részeként.

A kiválasztott szerepek bemutatásához az UML use-case diagramm típusát használom (<https://www.uml-diagrams.org/use-case-actor.html#business-actor> )



7. ábra: Modellezett üzleti szerepkörök

A szereplők bemutatása előtt a modellezés szempontjából minden szereplő az alkalmazott szereplő specializációja. A dolgozatban az alkalmazott általános szereplő funkciót megfeleltetem annak az általános felhasználónak, aki az SAP rendszert használhatja. Belépés után tranzakciót tud indítani, az SAP GUI képernyőkön DYNPRO keretrendszer által nyújtott funkciókat eléri, például felhasználó specifikus megjelenítési opciók (dátum, idő, nyelv) beállítása, nyomtatási funkciók, tranzakciók közötti navigáció, stb.

### Projektfelelős

Feladata a projekt végrehajtásának támogatása. A projekt egészéért felelős információk tárolásáért felel, például a projekt határideje, projekt státusza, projektekhez rendelt alkalmazottak kezelése, projekt feladatainak kezelése, különböző kimutatások elkészítése.

Felelős a projekt végrehajtásáért és biztosítja a projekt során szükséges ütemezési és újra ütemezési feladatokat.

### Projekt résztvevő

Feladata a projektben létrehozott feladatok végrehajtása. Egy, vagy több feladaton is dolgozhat, mely végrehajtás során az adott munka végrehajtásának lépéseiről tárol információt (a feladatot elkezdtem, a feladat késik, a feladat elkészült, stb.)

### Adminisztrátor

A rendszer technikai üzemeltetéséért felelős. Minden projekthez tartozó adathoz és technikai eszközhöz hozzáfér. Feladata a zavartalan működés biztosítása.

## Az alkalmazás funkciói

Az alkalmazás tervezet funkcióit szereplőnként mutatom be.

### Projekt felelős által elért funkciók



8. ábra: Projektvezető által elért funkciók

Projekt kezelése:

* Projekt létrehozása
* Projekt céljának kezelése
* Projekt státuszának beállítása (Kezdeti, Folyamatban, Késésben, Blokkolt, Elkészült)

Feladatok kezelése:

* Feladat létrehozása, törlése
* Feladat státuszának beállítása (Nem elkezdett, Folyamatban, Elkészült, Blokkolt)
* Feladatok ütemezése algoritmikus módszerekkel (CPM)

Résztvevők kezelése:

* Alkalmazott hozzárendelése egy projekthez
* Alkalmazott eltávolítása egy projektből

Kimutatások készítése

* Elvégzett feladatok listázása
* Blokkolt feladatok listázása
* Várható befejezés listázása

### Projekt résztvevő által elért funkciók



9. ábra: Projekt résztvevő által elért funkciók

TODO: leírni hasonlóan a projekt felelőshöz

### Adminisztrátor által elért funkciók



10. ábra: Adminisztrátor által elért funkciók

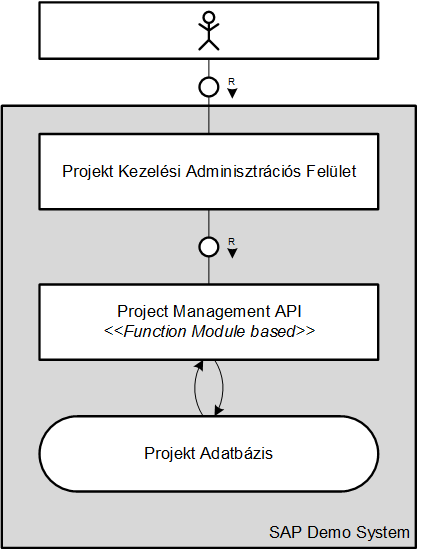
TODO: leírni hasonlóan a projekt felelőshöz

## Alkalmazás komponensei, fő felépítés

Ebben a fejezetben bemutatott a tervezett alkalmazás magas szintű architektúráját, fő rétegeit és komponenseit.

### Maga szintű áttekintés.

A modellezet szerepkörök az alkalmazást SAPGUI kliensen keresztül érik el, hasonlóan az általánosan használt SAP alkalmazásokhoz. A felhasználói felület egy funkciós modulokon alapuló project management API-n keresztül jelenít meg információkat, illetve kér adatmódosítást az adatbázis tartalmán. A projekt adatbázist az SAP rendszeren belül fogom létrehozni, a XYZ tárgy keretein belül tanult módszertanok alapján.



11. ábra: A projekttervezési szoftver felépítése

### Felhasználói felület

A megvalósított funkciótól függően a használt megjelenítési technológia lehet lista képernyő, ABAP riport, vagy ABAP dynpro. (todo: linkelni az irodalmat az órai jegyzetben is hivatkozottakra).

A komplex tervezés keretei között webes megjelenítési technológiával nem foglalkozom, de a későbbiekben ilyen technológián alapuló felhasználói felület kialakítására és integrálására lehetőség lesz.

### Projekt menedzsment API

A felhasználói felületek minden esetben az úgy nevezett projekt menedzsment alkalmazás programozási interfészhez (Application Programming Interface (API)) fog kéréseket küldeni.

Ennek oka, hogy később különböző, korszerűbb (webes) felhasználói felületek létrehozásakor ugyanazt az integrációs pontot tudjam használni és biztosítsam az alkalmazás konzisztenciáját. Ennek következménye, hogy az megjelenítési rétegben nem szabad direkt adatbázis lekéréseket és módosításokat végrehajtani, hiába van erre támogatott mód az ABAP nyelvben.

Az API valósítja meg az üzleti logikát. A jó fejleszthetőség és karbantarthatóság érdekében ezt a réteget több modulra és belső rétegre bontom, az alábbiak szerint.



12. ábra: Projekt menedzsment API belső rétegelése

### Adatmodell

Ezt még nem tudom pontosan mire gondolt Krisz, de szerintem 1-2 oldalt ide is lehet majd írni

Az alkalmazásban az alábbi információkat kell tárolnunk egy projekthez kapcsolódóan:

* Projektre vonatkozó információk
  + Azonosítója
  + Projekt célja
  + Projekt tervezett befejezési ideje
  + Státusza
  + Stb.
* Projekthez felelőse
* Projekthez rendelt alkalmazottak
* Projekthez tartozó feladatok és azok leíró információi
  + Feladat azonosítója
  + Feladat tervezett ideje
  + Feladat státusza
  + Befejezési ideje

Ez alapján az alábbi entitás-relációs diagrammot hoztam létre.



13. A tervezett alkalmas entitás-reláció diagramja

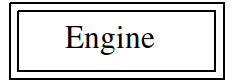
Ahhoz, hogy a projektmenedzsment solution megfelelően tudja kezelni a megjelenő projekteket, szükség van arra, hogy meghatározzuk a felmerülő szereplőket vagy entitásokat. Ezek modellezésére entitás-kapcsolat vagy ER-diagrammokat fogok használni.

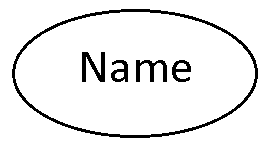
### Az ER-modell

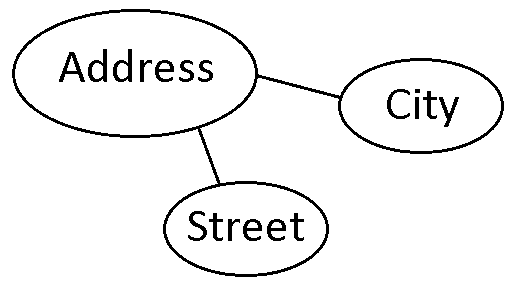
(<https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbazis_peldatar/ch02.html>  
Szűcs Miklós mesteroktató tantárgyjegyzetei)

Az Egyed-Kapcsolat modellt (Entity – Relationship vagy ER-modell) 1976-ban dolgozta ki Peter Chen számítógéptudós. A mai napig számos területen használatos, azonban leggyakrabban relációs adatbázisok tervezésénél alkalmazzák tervezési segédeszközként. Az ER-modell lényegében egy egyszerűsített szemantikai adatmodell, amely kizárólag strukturális, illetve elemi statikus integritási részt tartalmaz. Széles körben való elterjedését főleg egyszerűségének köszönheti, amely a grafikus jelölésrendszerét is jellemzi. A modell megalkotása során három alapvető objektumból építkezhetünk. Ezek az Egyedek, a Kapcsolatok, illetve az ezekhez tartozó Tulajdonságok.  
Az **egyed**ek tulajdonképpen olyan dolgok, amelyek a külvilág egyes részeitől elhatárolva önálló léttel bírnak, lényegében ezekről tárolunk információt.

Egyedekből kétféle egyedet különböztetünk meg, ezek

* az önmagukban is azonosítható **normál egyedek** (ilyenek pl. autó, dolgozó)  
  A normál egyedek jelölése egy egyszerű téglalap, benne az egyed nevével:
* az önmagukban nem, csupán normál egyedekhez való viszonyán keresztül azonosítható gyenge **egyedek** (ilyenek lehetnek pl. az autó motorja, vagy a dolgozó valamely hozzátartozója)

Az egyedekhez az Entitás-kapcsolat modellben különböző **tulajdonság**okat rendelhetünk. Ezeket egy egyszerű egyenes vonallal kötjük össze az adott entitásokkal. Az attribútumok típusai és a hozzájuk tartozó jelölések a következők

* a tulajdonságok legalapvetőbb típusa az egyértékű **normál tulajdonság,** jelölése egy egyszerű ellipszisen belül a tulajdonság nevével történik
* egy-egy egyed azonosítására a **kulcs attribútum** szolgál, amelynél az azonosító tulajdonságot aláhúzással jelezzük
* az **összetett tulajdonság** nevét onnan kapta, hogy az adott tulajdonság több további tulajdonságból épül fel. Ilyen pl. a lakcím, ahol a tulajdonság tovább bontható városra, utca névre és esetleg házszámra is
* a fentieken kívül megkülönböztetünk még **többértékű tulajdonság**otis, amelyet tipikusan arra használunk, ha adott tulajdonságból többet is hozzá tudunk rendelni egy egyedhez. Például egy személy hobbijainak nyilvántartására megoldást ad egy ilyen többértékű attribútum használata
* végül utolsó típusként a **származtatott tulajdonság**ot említhetjük, amelynek lényege, hogy más adatokból, esetleg tulajdonságokból számítható

A már ábrázolt egyedek között fennálló relációkat pedig a **kapcsolat**ok segítségével szemléltethetjük. A kapcsolatok ábrázolása egy rombusz segítségével történik, amelynek belsejébe a kapcsolat megnevezése kerül, illetve átellenes csúcsaiból kiinduló irányított egyenesek kötik össze az összekapcsolódó entitások téglalapjait.

A kapcsolatok csoportosításának számos fajtája van, ezek

1. Kötelező jelleg szerinti csoportosítás:

* **kötelező** egy **kapcsolat** abban az esetben, ha minden egyed-előforduláshoz kapcsolódik egy egyed-előfordulás a kapcsolatban, a szemléltetés duplázott irányított vonallal történik

5. ábra: termék egyed létezhet önmagában is, azonban rendelést kötelezően termékhez kell rendelni

* **opcionális** a **kapcsolat** akkor, amikor nem minden egyed-előforduláshoz kötődik egyed-előfordulás a kapcsolatban, egyszerű irányított vonalak

1. Kapcsolatban szereplő egyedek számossága szerinti csoportosítás:

* **egy-az-egyhez kapcsolat**ról beszélünk abban az esetben, amennyiben a kapcsolatban minden egyed-előforduláshoz csak és kizárólag egy egyed-előfordulás kötődik
* **egy-a-többhöz kapcsolat** esetén egy egyed-előforduláshoz több egyed-előfordulás is kapcsolódhat

6. ábra: egy osztályközösséget több tanuló alkoz, azonban egy tanuló csupán egy osztályhoz tartozik

* míg a **több-a-többhöz kapcsolat** nevét onnan kapta, hogy a kapcsolatban szereplő egyedek között mindkét irányban több egyed-előfordulás is engedélyezett

7. ábra: egy országhoz több folyó is tartozhat, ugyanakkor egy folyó több országon keresztül is futhat

A feladat szempontjából fontos, illetve az ER-modellben megjelenített szereplők:

##### 1. projekt

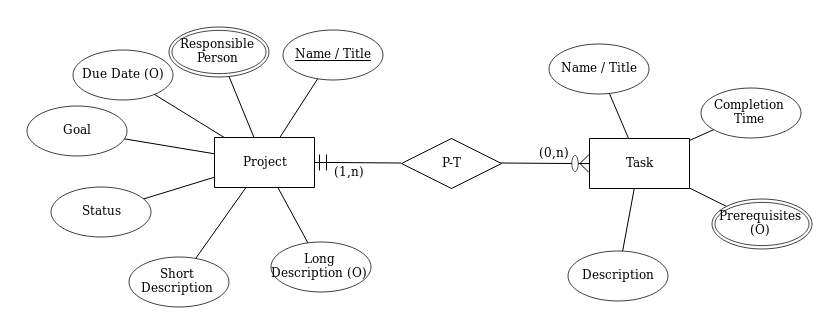
Mivel a szakdolgozatom témájában megjelölt szoftvermegoldás projektet kivitelezésében kíván segítséget nyújtani, így egyértelműen maguk a projektek adják az egyik legfontosabb entitáskört. Adminisztrációs szempontból minden projekt rendelkezik egy *névvel* vagy *címmel*. Van *határidejük* is, ezeket a vállalat politikája szerint léphetik át. Rendelkeznek továbbá egy *rövidebb* és/vagy *hosszabb leírással* is, illetve *státusszal* is, aszerint hogy milyen stádiumban létezik az adott projekt. Leggyakoribbak az aktív vagy folyamatban lévő, a befejezett, a még el nem kezdett, illetve átmenetileg felfüggesztett állapotok. Minden projekt valamilyen *célból* jön létre, így azt is érdemes rögzíteni.

A projekteknek vannak *felelősei(k)* is, illetve elvégzendő feladatokból állnak, ezek a feladatok adják a következő szerepkört.

##### 2. feladat

Minden feladatnak van egy *megnevezése*, illetve *leírása*, amelyben meghatározzuk, hogy mit kell elvégezni. Érdemes a projektet megfelelően kis számú feladatokra osztani, hogy egy-egy feladat könnyen elvégezhető lehet. Ugyanakkor a túl sok feladatra osztásnál az adminisztrációs és végrehajtási feladatok is túl sok időt vehetnek igénybe, így azt mondhatjuk, hogy ez sem igazán célszerű. A feladatok *végrehajtási ideje* is számos tényezőtől függhet, ezek közül az egyik, hogy mely gépen vagy milyen erőforrás segítségével hajtódnak végre. Abban az esetben, ha a már megfelelően definiált és végrehajtásra váró feladatok több erőforráson is végrehajtódhatnak, úgy érdemes a kivitelezéssel járó időket rögzíteni, tárolni. Erre az egyik legegyszerűbb és legkézenfekvőbb megoldás egy mátrix, amelyben az oszlopok jelölhetik a végrehajtásra váró feladatokat, a sorok pedig a különböző erőforrásokat, legyenek azon mechanikus vagy emberi jellegűek.  
Speciálisan az én projektmenedzsment megoldásomban egyetlen erőforrás áll majd rendelkezésre, ez fogja végrehajtani a feladatokat, így elegendő csupán letárolni az egyes feladatokhoz tartozó végrehajtási időket.  
Az egyes feladatoknak lehetnek *előfeltételeik*, azaz egészen addig nem indíthatók el vagy hajthatók végre, amíg az előfeltételben definiált állapot fent nem áll. Ez az én megvalósításom szerint egy másik feladat végrehajtásának teljesülése lehet.

A fenti 2 entitásból, illetve a köztük lévő relációból az alábbi ER-modellt alkottam meg:

x. ábra: Az projektmenedzsment solution ER-modellje (prototípus)

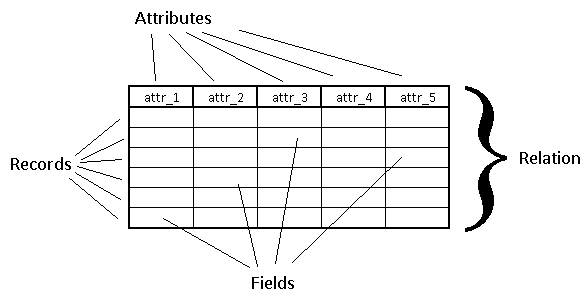
A fenti ábrából könnyen látható a projektek, illetve a hozzájuk kapcsolódó feladatok relációja. A projektmenedzsment rendszerben elsőként a projekteket, majd a feladatokat hozzuk létre. Mivel elsőként a projektek jönnek létre, úgy konvencionálisan létezhet projekt feladat nélkül, azonban feladat önmagában nem, azt hozzá kell rendelni egy már meglévő projekthez.

A vázolt ER-modell ábrából pedig már könnyen meghatározhatók az adatbázisbeli letároláshoz szükséges táblák a megfelelő attribútumaikkal.

### A relációs séma

(Szűcs Miklós mesteroktató tantárgyjegyzetei  
https://hu.wikipedia.org/wiki/Rel%C3%A1ci%C3%B3s\_adatmodell)

A relációs séma vagy adatmodell E.F. Codd nevéhez fűződik, aki 1970-ben publikált *A Relational Model of Data for Large Shared Data Banks* cikkében mutatta be azt. Kulcsfogalomként jelenik meg a matematikából már ismert reláció fogalma, amely lényegében a Descartes-szorzat részhalmaza. A Descartes-szorzat pedig valójában önálló egységek vagy egyedek egymás mellé rendelését jelenti. A relációs adatmodell erre a típusú relációra épül. Adatmodellként meghatározza az adatszerkezeteket és az adatmodelleken értelmezett műveleteket is. A relációs adatmodell csupán logikai szempontból határoz meg bizonyos dolgokat, így a tényleges hardveres megvalósítással, mint például a memóriában való tárolással, az adatok onnan ki-, és bemozgatásával, illetve az információ tárolásának egyéb módszereivel nem foglalkozik.   
Adatbázisok kialakításánál elterjedt megoldás relációs sémára épülő vagy relációs adatbázis használata. A relációs adatbázisban tulajdonképpen relációkat tárolunk el. Egy-egy ilyen reláció egy adott egyedcsoportból származó, hasonló egyedek tulajdonságait tároló táblázatot jelent, ahol a relációt gyakrabban tábla vagy táblázat, a tábla sorait rekordok, az oszlopokat pedig tulajdonság vagy attribútum néven említjük. Egy táblázaton belül adott rekordhoz tartozó, valamely tulajdonság alatt található értéket pedig eleminek tekintünk és mezőnek nevezzük.



8. ábra: egy relációs adatbázisban alkalmazott táblázat felépítése az egyes részek megnevezéseivel

A fentebb tárgyaltak alapján tehát láthatjuk, hogy a relációs adatbázis a 4. ábrán is látható szerkezetű táblázatokból áll. A táblázatokat rekordok sorai építik fel, amelyek a logikailag összetartozó és egységként kezelhető mezők együtteseit jelentik. A mezők adják az adatbázis struktúra atomi egységét, ezekből épülnek fel a rekordok.

Az adatok, illetve a kapcsolatok ábrázolására a fentiek alapján tehát egy séma-attribútum-reláció hármast alkalmaznak, amelyben a séma jelenti a táblázat felépítését vagy szerkezetét, az attribútumok vagy tulajdonságok adják az oszlopokat, illetve a reláció maga jelenti a táblázatot, ahogyan azt fentebb már ismertettem.

A mezők jellemzése során lehetőségünk van megadni az integritási feltételeket, illetve a mező domain-ét. A domain jelenti a mező típusát és kijelöli azt az értelmezési tartományt, amely értékkészletből a mező értékeket vehet fel, valamint a végrehajtható műveletek halmazát is megadja. Ilyen alapvető domainek a

* karakteres mező, jelölése: C(x), ahol x a mező hosszát kijelölő természetes pozitív szám
* numerikus vagy számértékű mező; Number(j,k), ahol j a tizedesvesszőtől balra, k pedig az attól jobbra elhelyezkedő helyiérték-sorozat hosszát adja meg
* dátum; date, ahol további formai követelményeket adhatunk meg

A mezők vagy cellák helyére csupán egy érték kerülhet, így az ER-modellből ismert többértékű vagy összetett tulajdonság közvetlenül nem szerepelhet egy rekordon belül, ezek tárolása egy másik táblában történik speciális mezők, ún. kulcsmezők és kapcsoló mezők segítségével.

A rekordok szerkezetét sémákban tároljuk, ezek a tábla nevét, továbbá a mezők neveit, típusait vagy domaineit, illetve integritási feltételeit tartalmazza.

A relációk egyik jellemző tulajdonsága a reláció egyedszáma, amely a hozzá tartozó rekordok számát jelenti. Fontos kikötés, hogy egy relációban nem helyezkedhet el két teljesen identikus rekord. Egy táblában ugyanakkor két ugyanolyan névvel ellátott tulajdonság vagy oszlop sem megengedett. A relációk másik jellemzőjét adja a reláció foka, amely az attribútumok számát adja meg.

Az itt látható példában a reláció fokszáma 2, ugyanis két tulajdonságot vagy attribútumot kívánunk tárolni, míg az egyedszáma 4, mivel négy rekord található benne.

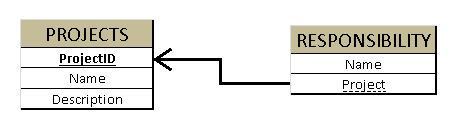
9. ábra: példa reláció, amelyben összesen 2 attribútum kap értéket, összesen 4 rekord esetében

A kapcsolatok szemléltetése indirekt módon történik, azaz nem külön elemként tároljuk őket a modellben, hanem az összefüggő relációkban egy-egy kijelölt, speciális szerepű attribútum értékegyezősége fogja adni a kapcsolatot. Ezeket az attribútumokat az egyik relációban elsődleges kulcsként, a hozzá kapcsolódóban pedig kapcsoló kulcsként említjük. A kapcsolatok számossága, illetve kötelező jellege határozza meg azt is, hogy a kapcsoló kulcs értéke adott relációban ismétlődhet-e, illetve szükséges-e kapcsolótábla használata. Ez utóbbi lényegében egy olyan relációt jelent, amelyben csupán elsődleges és kapcsoló kulcsok találhatók.

A relációk megadására számos módszert alkalmaznak, ezek közül a legelterjedtebbek a

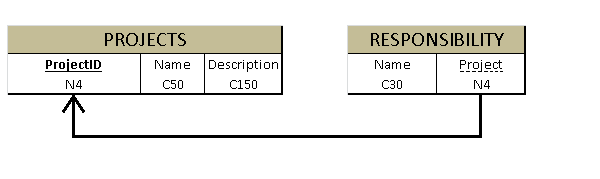
1. Bachman-diagram:

Nevét Charles Bachmanról kapta, aki az 1960-as és 1970-es években A. P. G. Brown-al együttműködve dolgozott ki egyfajta Adatstruktúra grafikont, amely később Bachman-diagram néven vált ismertté. Jellegzetességei, hogy a relációkat téglalapokként tárolja, legfelül a tábla nevével, amelyet csupa kapitális betűvel ír. A név után következik a tényleges tábla szerkezet, legfelül az elsődleges kulcsokkal, aláhúzva. Ezeket követik a másodlagos mezők vagy attribútumok, végül pedig a kapcsoló kulcsok szaggatott aláhúzással. Magukat a kapcsolatokat pedig irányított nyilakkal szemlélteti.



1. Struktúra ábra:

10. ábra: példa a Bachman-diagramra. A PROJECTS táblában található ProjectID elsődleges kulcs és a RESPONSIBILITY táblabeli Project névvel ellátott kapcsoló kulcsok értékegyezősége adja a kapcsolatot projekt és felelőse között

A Bachman-diagrammal ellentétben az egyes mezők felsorolása nem függőleges, hanem vízszintes irányban történik. Továbbá a mezők domainjei vagy típusai is feltűnnek a szemléltetés során.

1. Relációs séma leírás:

A relációs séma leírás során szöveges formában határozzuk meg az egyes relációkat a hozzájuk tartozó attribútumokkal egyetemben. Ugyanakkor az egyes mezőket érintő integrációs feltételek vagy a domainek nem kerülnek rögzítésre. A fenti 2 relációt leíró relációs séma leírás a következőképpen nézne ki:

**PROJECTS** [ProjectID, Name, Description]

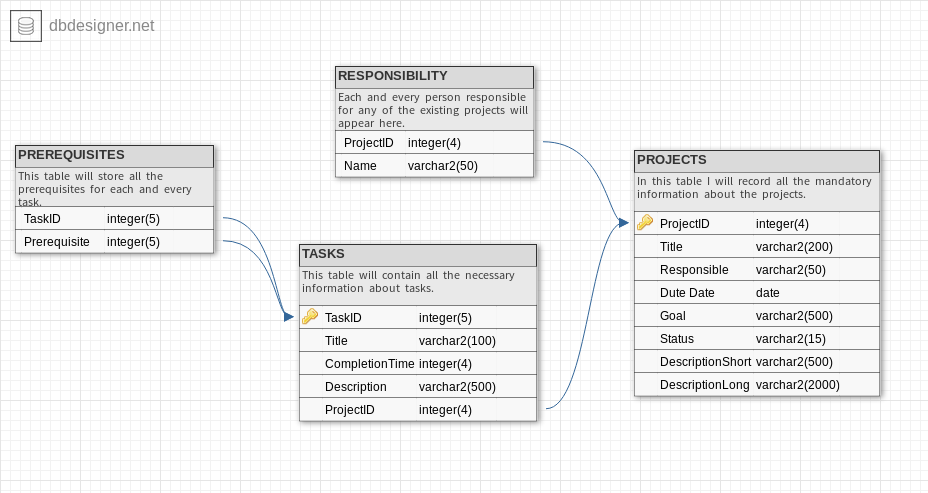
**RESPONSIBILITY** [Name, Project]

Végül érdemes pár mondatban megemlíteni a fentebb már többször is említett adatintegritásról. Az adatintegritás lényegében a bevitt adatok helyességét, jóságát jelenti, azaz hogy megfelelnek-e bizonyos elvárásoknak, illetve előírásoknak. Az adatintegritást értelmezhetjük

* mező szinten, ekkor az egyes mezőkbe kerülő adatokat vizsgáljuk. Megkötés lehet, hogy az értéke csak bizonyos tartománybeli értékeket vehet fel, például a húsznál kevesebb pozitív egészeket vagy az is, hogy kötelező legyen értéket adni a mezőnek. Előre definiált formátumot is megadhatunk, ez is az adatintegritáshoz tartozik.
* rekord szinten, ekkor a logikailag összetartozó mezőkből álló rekordokról kell eldöntenünk, hogy helyesek-e vagy sem, azaz megfelelnek-e az adatintegritás szabályainak. Leggyakoribbak azok az esetek, amikor több mező értéke kapcsolatban áll egymással, ezekre adhatunk feltételeket. Például ha egy autó évjárata 2000 utáni évszám, akkor az értéke legyen nagyobb 1000000 forintnál.
* reláció szinten, amikor az egész táblára nézve ellenőrizzük az adatok integritását. Tábla szintű megkötés lehet, hogy a kulcsmezőként használt attribútum értékeinek teljesen egyedinek kell lenniük és csak egyszer fordulhatnak elő egy reláción belül.
* adatbázis szinten, ehhez több reláció mezőinek értékének vizsgálata szükséges. Ilyen adatintegrációs megkötés lehet például, hogy a PROJECTS relációban található, egy adott projekt státusza addig nem vehet fel *Running* értéket, amíg nem rendeltünk hozzá egy felelőst a RESPONSIBILITY táblából, vagy a projekt feladatainak precedencia szerinti sorrendjében az utolsó feladat indítási ideje sohasem előzheti meg az első feladat indítási idejét

A relációs séma széles körben való elterjedését többek között egyszerű felépítésének, a műveletek, illetve lekérdezések hatékonyságának, valamint a kapcsolatok rugalmas rendszerének köszönheti.

A fentebb elkészített ER-modell terv, illetve a relációs adatbázis-tervezésről szerzett ismeretek alapján az alábbi sémákat készítettem el:

x+1. Ábra: a projektmenedzsment megoldás magját adó relációs adatmodell

Az ábrán látható relációs adatmodell a következő táblákból épül fel:

##### 1. a PROJECTS tábla

Mint ahogyan az elnevezés is mutatja, ebben a táblázatban fognak szerepelni a solution által kezelendő projektek, a hozzájuk tartozó adminisztrációs adatokkal együtt.  
Elsősorban mindegyik rekordhoz tartozni fog egy elsődleges kulcsot reprezentáló, 4 karakter hosszúságú integer értéket felvevő mező, amely a *ProjectID* nevet kapta. Az értékeket kézzel nem lehet megadni, inkrementálisan kap értéket minden egyes úgy projekt azonosítója. Emiatt az érték majd egyedi (unique) lesz.  
A projekt megnevezését vagy címét egy 200 karakter hosszúságú sztringként tárolom. Itt is megkötés, hogy minden projekt különböző névvel szerepeljen, azaz a mezőnek egyedinek kell lennie, továbbá kötelezően meg kell adni azt. Az attribútum az egyértelműség kedvéért a *Title* nevet kapta.  
A határidőket egy egyszerű date formátumban tárolom le a *DueDate* mezőben. A megvalósítás szempontjából ez üresen is hagyható. Később az optimális végrehajtási sorrend meghatározásánál szükség lehet a határidők időegységre (nap) való konverziójára, hogy a feladatokhoz tartozó végrehajtási időkkel egy matematikai modellben tudjuk azokat számolni.  
A projekt végső céljának meghatározására szintén egy hosszabb szöveges (sztring) mező áll rendelkezésre.  
Az adott pillanatban a projektre jellemző stádiumot vagy állapotot a *State* mezőben adjuk meg. A tárolásnál ismét szöveges értéket adunk meg. (kikötés? → addig nem indítható el egy projekt [State nem lehet »ongoing«], amíg legalább egy felelős nincs beállítva a számára.)  
Végül a projektről pár sorban rövid leírást adhatunk, melynek maximális hossza 500 karakter lehet és a *DescriptionShort* mező manipulálásával tudjuk a leírást változtatni.   
Utolsó attribútumként jelenik meg a *DescriptionLong* nevű, amelyben további kiegészítő információkat adhatunk meg az adott projektről, abban az esetben, ha a rövid leírás nem lenne elegendő. Ugyanakkor ezen mező kitöltése opcionális.

##### 2. a TASKS tábla

A TASKS táblában tárolom az egy-egy feladathoz tartozó rekordokat, amelyek ezen feladatokhoz tartanak nyilván adatokat.  
Azonosító mezőként jelenik meg minden feladathoz a kulcs értéket megvalósító, 5 számjegy hosszúságú integer formájában megjelenő *TaskID* attribútum.   
Kötelezően kitöltendő a feladat címét vagy megnevezését maximum 100 karakter hosszan tároló *Title* mező, ami a TASKS táblára nézve egyedi, azaz nem létezhet kettő feladat ugyanazzal a címmel.  
A feladatok végrehajtási sorrendjének meghatározásának előfeltétele, hogy ismerjünk azok végrehajtási idejét. Jelen esetben ezek tárolására egy maximálisan 4 hosszúságú integer mező áll rendelkezésre *CompletionTime* néven.  
Mivel a fentebb említett Title mezőre azt a kikötést tettem, hogy egyedinek kell lennie, így ugyanez áll fenn a leírásokat tartalmazó *Description* attribútum esetén is, hiszen amennyiben 2 feladat leírása megegyezhetne, úgy az azt jelentené, hogy a két feladat teljesen identikus. Azonban ugyanazon feladat két helyen való tárolása redundáns, nincs rá szükség.  
Mint már a modell tervezése során leírtam, egy-egy feladat csak egy projekthez tartozhat, illetve feladat nem létezhet projekt nélkül, mindenképp valamelyik projekt részeként valósulhat csak meg. Ezért van szükség idegen kulcs alkalmazására, amely majd ezt az összeköttetést megvalósítja projekt és feladat között. Az idegen kulcs a PROJECTS táblában lévő ProjectID elsődleges kulcsra mutat és az egyszerűség kedvéért ugyanazt a nevet viseli, illetve a használt adattípus szintén egy 4 hosszúságú integer érték.

##### 3. a RESPONSIBILITY tábla

A fentebb vázolt ER-modellben a projektek mellé rendelt felelősöket reprezentáló *Responsible Person* tulajdonság többértékű. Ahhoz, hogy ezt tárolni tudjuk az adatbázisban, szükség van egy újabb táblára, amely az egyes projektekért felelős személy(eke)t tárolja. A megvalósítás egy egyszerű, mezőpárokként megvalósuló rekordok táblázataként történik.  
Ebben az első mező a már korábban, a TASKS tábla esetén szemléltetett idegen kulcsként jelenik meg, és elnevezését szintén a PROJECTS táblabeli ProjectID azonosító után kapta. Ezzel beazonosítjuk, hogy melyik projekthez rendelünk felelőst.  
A második mezőben pedig magának a felelősnek a megnevezése tárolódik a *Name* attribútumban, szöveges formátumban, legfeljebb 50 karakter hosszan.   
Amennyiben egy projekthez csupán egy felelős tartozik, úgy a RESPONSIBILITY táblában mindössze egy rekord esetén lesz egyezőség az elsődleges kulcs ProjectID és az idegen kulcs ProjectID között. Ugyanakkor, amennyiben egy projekthez több felelős is tartozik, úgy több rekord esetén is meg fog egyezni az idegen kulcs ProjectID értéke. Ezen rekordokra fennáll az a feltétel, hogy a felelős nevét tartalmazó Name mező értéke nem egyezhet meg, ugyanis ez újabb adat duplikációval járna.

##### 4. a PREREQUISITES tábla

Végül ahhoz, hogy az egyes feladatokhoz tartozó előfeltételeket el tudjuk tárolni, szükség van egy külön táblára, amely csak ezeket adminisztrálja. Hasonlóan a RESPONSIBILITY tábla felépítéséhez, ebben a táblában is mezőpárok formájában jelennek meg a rekordok.  
A lényeges különbséget az adja, hogy a fentebb részletezett táblával szemben itt mindkét mező idegen kulcs, méghozzá a TASKS táblában szereplő TaskID elsődleges kulcsra mutató idegen kulcsok.  
Elsőként szerepel az adott feladat azonosítója, másodikként pedig a feladat előfeltételét adó végrehajtandó feladat azonosítója.  
Feladatok létezhetnek előfeltétel nélkül, ebben az esetben adott feladathoz kötődő rekord nem is kerül ebbe a táblába, egy, illetve több előfeltétellel is. Ugyanakkor feladat nem lehet önmaga előfeltétele, hiszen ez holtponthoz vezet.

## Tervezett felhasználói felület

Na, szerintem ide jöhet úgy 2-3 oldal simán. Elméletben ide kerül be hogy milyen képernyők lesznek, honnan mi indítható.

# Megvalósítás

## Adatbázis modell megvalósítása SAP adatbázisban

## Üzleti logika rétege (funkciós modulok)

Hát ide is kell majd Krisz segítsége. Hirtelen most nem sok minden jut eszembe az üzleti logikáról. Bár a funkciós modulokról igen, valószínűleg ide kerülnek az egyes funkciókat (feladat projekthez rendelése, új erőforrás felvétele, új projekt beillesztése, új szereplők felvétele, meglévők menedzselése, stb) megvalósító modulok. Amennyiben tényleg erről van szó, úgy 2-3 oldalt ide is lehet majd írni, jó sok mindent tudnia kellene a szoftvernek.

## Felhasználói felület (SAPGUI képernyők, navigációk, stb)

Szerintem az ehhez tartozó rész nagyját még a tervezésnél le fogom írni és mivel konkrét implementációt nem igazán lehet használni, így ide olyan 1, max 2 oldal fog jönni. Bár képekkel kiegészítve ez is kitehet akár többet is.

# Melléklet

## Ábrajegyzék

[1. ábra: A vállalaton belül szereplő főbb tevékenységtípusok 5](#_Toc514688565)

[2. ábra: A Weiss és Wysocki féle ötfázisú projekt-életciklus modell 6](#_Toc514688566)

[3. ábra: Új iroda létesítésének egy lehetséges WBS vázolása 7](#_Toc514688567)

[4. ábra: Felelősségi és kompetencia mátrix lehetséges felépítése 8](#_Toc514688568)

[5. ábra: Példa CPM-el meghatározott hálótervre 8](#_Toc514688569)

[6. ábra: Példa MPM-el készített hálótervre 9](#_Toc514688570)

[7. ábra: Modellezett üzleti szerepkörök 12](#_Toc514688571)

[8. ábra: Projektvezető által elért funkciók 13](#_Toc514688572)

[9. ábra: Projekt résztvevő által elért funkciók 15](#_Toc514688573)

[10. ábra: Adminisztrátor által elért funkciók 15](#_Toc514688574)

[11. ábra: A projekttervezési szoftver felépítése 16](#_Toc514688575)

[12. ábra: Projekt menedzsment API belső rétegelése 17](#_Toc514688576)

[13. A tervezett alkalmas entitás-reláció diagramja 19](#_Toc514688577)

## Irodalomjegyzék

[1] – A projekt fogalma, a projektek csoportosítása

http://centroszet.hu/tananyag/projektmenedzsement/12\_a\_projekt\_fogalma\_a\_projektek\_csoportostsa.html

[2] – A projektek felépítése

Dr. Garaj Erika (2012): Projektmenedzsment

Nagy Zsolt: *Projektmenedzsment alapjai*

Jesse Santiago & Desirae Magallon (2009): *Critical Path Method*

Herold Kerzner (1979): Project Management: *A Systems Approach to Planning, Scheduling and Controlling*, John Wiley and Sons, Hoboken

Dr. Daróczi Miklós (2014): *Projektmenedzsment*, John Wiley and Sons, Hoboken

Tarcsi Ádám, Molnár Bálint (2014): *Vállalatirányítási rendszerek gazdaságinformatikai megközelítésben*, Eötvös Lóránt Tudományegyetem, Informatikai Kar

Steffen Karch, Loren Heilig (2005): *SAP NetWeaver Roadmap.* Galileo Press, ISBN 1-59229-041-8

Horst Keller, Joachim Jacobitz (2003): *ABAP Objects The Official Reference,* SAP PRESS

[2] - Projektek ismérvei

Szerző2 (1999): Cím2, Kiadó, Hely