Debreceni Egyetem, Informatikai Kar, Információ Technológia Tanszék

Az *ATEV* partnerinformációs rendszere

DIPLOMAMUNKA

Témavezető:
Vágner Anikó
egyetemi tanársegéd

Készítette:
Nagy István
programtervező matematikus

Debrecen,
2010

1 Bevezetés

Napjainkra egyre több teret hódít magának a nagy, vállalati rendszerek mellett a kisebb célszoftverek csoportja is. Egyre többen látják be azt, hogy nem lehet minden esetben ráhúzni a vállalati szoftverrendszer működését egy-egy adott részfolyamatra, pontosan ezen nagy szoftverek általános volta miatt. Ezen rendszerek nincsenek felkészítve speciális esetekre, csupán az alap, minden vállalatra teljesülő funkciók szerepelnek bennük. Ugyan megfelelő plusz összegért szívesen integrálnak az adott cég struktúrájába illeszkedő modulokat, viszont ezek a plusz modulok a verzióváltással elévülnek, újraírásra szorulnak, ami további idő és költségvonzatot jelent a cégek számára.

A célszoftvereknek megvan a sokszor behozhatatlan előnyük, hogy teljesen az elvárásokhoz, a cég szája íze szerint készül el. Ebből fakadóan kimondottan az adott területre szükséges működéssel bír, nem tartalmaz felesleges adatokat, melyek nehezítik, lassítják a program használatát.

Diplomamunkám témája egy ilyen célszoftver tervezése és elkészítése, melyet az ATEV Zrt. Nyersanyag- és fuvarszervezői illetve a Logisztikai munkatársak fognak használni. Dolgozatomban kiemelten fogom tárgyalni egy ilyen rendszer fejlesztéséhez szükséges tervezői munkát, továbbá, hogy a fejlesztés menete során milyen szempontokat, irányelveket vettem figyelembe.

Diplomamunkámban kifejtem azt is, hogy miért is volt szükséges a már meglévő vállalati rendszer mellé egy célszoftvert is készíteni, milyen előnyei vannak ennek az új szoftvernek és hogy milyen hiányosságait váltja ki a már meglévő rendszernek.

2 Az ATEV partnerinformációs rendszere

2.1 A rendszer elkészítésének háttere

2.1.1 Cégtörténet

Az ATEV Fehérjefeldolgozó Zrt jogelődjét, az Állatifehérje Takarmányokat Előállító Vállalatot (ÁTEV) az Országos Közellátási Hivatal 1949. július 1-évl alapította, az országban keletkező állati hulladékok begyűjtésére, illetve ártalmatlanítására. A megalapított vállalat az állati hulladékok kezelését végző telepeit 1950-ben kezdte meg fejleszteni, mely a társaság történetét mind a mai napig végig kíséri.

Az 1949-ben alapított állami vállalat 1993. szeptember 30-i hatállyal – 100 részvénytársasággá alakult, 878 MFt jegyzett–%-os állami tulajdonban álló tőkével. Az állami tulajdonosi jogokat a földművelésügyi és vidékfejlesztési miniszter gyakorolja. A gazdasági társaságokról szóló 2006. évi IV. törvény előírásainak megfelelően a részvénytársaság jelenlegi elnevezése: ATEV Fehérjefeldolgozó Zártkörűen Működő Részvénytársaság (ATEV Zrt).

2.1.2 A cég tevékenységei

Az állattenyésztés az állatvágás, a húsfeldolgozás és az élelmiszeripar melléktermékei és hulladékai (elhullott állatok, vágóhídi melléktermékek és hulladékok, húsfeldolgozási maradékok, konyhai hulladékok, romlott állati eredetű élelmiszerek, húsipari szennyvíziszapok stb.) alkotják társaságunk nyersanyagait, melyeket a jogszabályok – kezelési követelményeiktől függően –három kategóriába sorolnak.

Az állati melléktermékek kezelésének teljeskörűségét az alábbi módokon biztosítjuk:

1. A takarmányozásra, ipari felhasználásra alkalmas (3. osztályba sorolt) állati melléktermékekből takarmányt, illetve takarmány-alapanyagot, valamint takarmányzsírt , illetve ipari zsírt gyártunk.

2. A takarmánygyártásra ipari felhasználásra alkalmatlan (3. osztályba sorolt) melléktermékeket vagy jogszabályok szerint abból kizárt (2. osztályba sorolt) állati hulladékokat biogáz-, illetve komposzttelepen történő továbbkezelés céljából előkezeljük. Ezekből energiát, illetve termékjellegű anyagot lehet előállítani (lebontási maradék, komposzt).

3. A hasznosíthatatlan (1. osztályba sorolt), vagy az előzőekben felsorolt kezelésre gazdaságilag alkalmatlan állati hulladékokat égetésre előkezeljük liszt formátumra.

A 3. kategóriájú alapanyagból társaságunk csont-húslisztet, vérlisztet, valamint állat zsírt állít elő.

2.1.3 A cég szolgáltatásai

Az **ATEV Fehérjefeldolgozó Zrt.** az ország egész területén keletkező állati eredetű hulladékok, melléktermékek összegyűjtését és ártalmatlanítását végzi. A társaság megalapítása óta látja el feladatát, amely szorosan kapcsolódik az állategészségügyhöz és a környezetvédelemhez. Az ATEV Zrt. gyáraiban már az EU-hoz történő csatlakozást megelőzően is olyan technológiával történt az állati eredetű hulladékok feldolgozása és ártalmatlanítása, amely megfelel mind az EU, mind a hazai rendeletek előírásainak. Minden gyár rendelkezik EU regisztrációs számmal.

Tevékenysége két fő csoportra osztható:

- hulladékkezelési szolgáltatás,
- késztermék előállítás és értékesítés.

1. A **hulladékkezelési szolgáltatás** magában foglalja az állati hulladék szelektív begyűjtését, szállítását és feldolgozását.

   A szolgáltatásunkat az alábbi partnerek veszik rendszeresen igénybe:

   - vágóhidak, hús- és baromfifeldolgozók,
   - állattartó telepek,

- önkormányzatok,
- állati eredetű élelmiszert feldolgozó üzemek,
- élelmiszerforgalmazó áruházláncok, vendéglátóipar, étkeztetést végző intézmények stb.

2. **Késztermék:** az EU 1774/2002 számú rendeletében meghatározott 3. kategóriájú alapanyagból csont-húslisztet, vérlisztet, valamint állat zsírt állít elő.

A csont-húsliszt felhasználási területei:

- kedvtelésből tartott állatok eledelének és
- nem élelmiszertermelő állatok részére készített takarmány alapanyagaként.

Állati zsír felhasználási területe:

- élelmiszertermelő állatok részére előállított takarmány alapanyagként
- ipari célra

A társaság által megtermelt zsírból készül az ATEV zsírpor termék, amelynek összetétele 40% takarmányzsír és 60% extrudált kukoricapehely-dara. Ez kiválóan alkalmas baromfi-, valamint sertéstakarmányok energiaszintjének biztosítására valamint növelésére.

2.2 A rendszer feladata

2.2.1 A központi rendszer

Mivel egy állami cégről van szó, melynek az egész országban vannak üzemei, telephelyei, így szükség van egy központi rendszerre, mely az összes kirendeltség adataival bír. Erre a feladatra a ’One World’ vállalatirányítási rendszert alkalmazza a vállalat. A One World-ben minden alapmodul rendelkezésre áll a logisztika disztribúció, a pénzügy számvitel, a termelésirányítás, illetve projektmenedzsment témaköreibe csoportosítva. A modulok lego-szerűen egymásra építhető többcsatornás funkcionalitásból építkeznek, melyek paramétereivel irányítva a legbonyolultabb üzleti modell is áttekinthetően kezelhető.

2.2.2 Az ATEV partnerinformációs rendszere

A központi rendszerrel való probléma abból adódik, hogy a vállalatnál még a bevezetéskori (1999-es) verzió van jelenleg is, és nincs kilátás a verzió váltásra, a speciális tevékenységből adódó kiegészítések miatt. Ugyanis az újabb verzió bevezetése esetén az eddig a régi verzió alatt megírt, speciális programegységeket az új verzióban is implementálni kellene, ami horribilis pénzösszeget emésztene fel.

Az általam fejlesztett rendszert alapvetően a nyersanyag-, és a fuvarszervező munkakörben levők használnák, tehát nem egy globális, mindent területet lefedő rendszerről van szó. Ez a rendszer nem végez számlázást, kizárólag a partner és felrakóhelyek adataiból szolgáltat testreszabott információt, jelentést a munkakör dolgozói részére, megkönnyítve a partnerrel való kapcsolattartást, a szállítások szervezését, gyors és pontos munkavégzést.

Ezen kívül a központi rendszer alábbi hiányosságai, nehézségei indokolták ennek a rendszernek a kifejlesztését:

- A rendszer együtt kezeli az aktív és inaktív partnereket
- Sok információt nem tartalmaz a rendszer és nincs is ezek berögzítésére lehetőség
- A rendszer kezelése bonyolult, egyáltalán nem felhasználóbarát és nem szolgálja a gyors munkavégzést
- A különböző felhasználói igényeket nehéz összeegyeztetni a bonyolult jogosultságkezelése miatt
- Az archiválás nem megoldható a programban
- Ebből adódóan a hatalmas adatbázis miatt lassú az adatelérés, illetve a megfelelő információk eléréséhez nagyon sok ismerettel kell rendelkezni
- Nincsenek előre összeállított lekérdezések, minden forrás táblát, lekérdezési feltételt és egyéb információt minden alkalommal meg kell adni
- Nehéz biztonsággal kezelhető, az eredeti célnak megfelelő adatokat lekérdezni a rendszer komoly ismerete nélkül

- A jogszabályok állandó változása miatt a rendszer használata tovább bonyolódik, nagyon sok olyan adatot tartalmaz, amelyre az adott munkakörben nincs szükség
- Mivel az összes telephely adatait tartalmazza, ez is lassítja az adatelérést
- Ebben a munkakörben csak adatlekérdezés történik a rendszerből, ami nagyon nehézkes, lassú és testre szabhatatlan, nem elégíti ki a felhasználói igényeket

Illetve a várható előnyök, amelyek az általam fejlesztett rendszer használatából adódnak:

- Ez a rendszer kimondottan csak a nyersanyagszervezői illetve fuvarszervezői munkakörhöz kapcsolódik
- Ebből adódóan teljes körűen testre szabható
- Csak azokat az adatokat tartalmazza, melyre az ebben a munkakörben dolgozóknak szüksége van
- Előre elkészített lekérdezések végezhetőek
- Az adatelérés gyorsasága jelentős mértékkel jobb
- Olyan speciális információk rögzítését biztosítja, melyek a fő rendszerben nem megoldottak

A fejlesztett rendszer egyetlen hátránya, hogy nem lesz képes kommunikálni a fő rendszerrel, és emiatt külön adatbevitelt igényel. (A fő rendszerrel való kommunikáció megoldása túlnyúlik a diplomamunka követelményein, de később, ha igény lesz rá, megvalósításra kerülhet)

2.2.3 Az ATEV partnerinformációs rendszerének felépítése

- Partner egység
- Felrakóhely egység
- Szerződés egység
- Fuvarszervező egység
- Telefonkönyv és címjegyzék egység
- Adminisztrációs egység

2.2.4 Az egységek feladatai

2.2.4.1 Partner egység

Feladata: a vállalattal üzleti kapcsolatban álló partnerek általános adatainak kezelése. Többek között a partner neve, címe, kapcsolattartó, elérhetőségek jelennek meg itt. Ebből az egységből indul ki majdnem az összes többi egység.

2.2.4.2 Felrakóhely egység

A felrakóhely egység a program ’lelke’. Ehhez az egységhez kapcsolódik a legtöbb tárolandó adat. Az ebben az egységben tárolt adatok képezik a fuvarszervező egység bázisát is. Ebben az egységben találhatók meg a partner telephelyeinek adatai, többek között a címe, a távolsága, az általa átadott anyagok jellege, típusa, az elszállítás jellege, melyik járat hozza be az anyagait, a behozott anyagmennyiséget ki és hol méri le, illetve egyéb fontos adatok is itt jelennek meg.

2.2.4.3 Szerződés egység

Ez az egység csak közvetlenül a szerződéshez szükséges adatokat kezeli, tárolja, elkülönítésének célja a mindennaposan használt adatok gyorsabb elérése, illetve a könnyebb áttekinthetőség.

A szerződések felépítéséből adódóan van külön a partner egységhez és külön a felrakóhely egységhez kapcsolódó szerződési egység is. Míg a partner egységhez kapcsolódó rész a szerződés törzsét alkotja, addig a felrakóhely egységhez kapcsolódó rész a szerződés mellékleteként jelenik meg.

2.2.4.4 Fuvarszervező egység

Ebben az egységben tervezhetőek meg az elkövetkezendő szállítások, napokkal előre is. Az egység kialakításának célja a pontosabb fuvarterv készítése, illetve az út terv pontosabb elkészítése (minél rövidebb utat tegyenek meg a gépkocsik, üzemanyag megtakarítás céljából)

A szállítás tervezése két lépcsős folyamat:

1. Minden felrakóhelynél lehetőség van berögzíteni fix szállítási napokat (H, K, stb.) télre/nyárra. Ezeket a fixen berögzített beállításokat egy listában összesítjük, amit a fuvarszervező végigellenőriz, és jóváhagyja vagy elutasítja az egyes szállításokat. A jóváhagyott szállításokból létrehozunk egy listát és azt eltároljuk.
2. A másik lehetőség kézi rögzítés. A megadott felrakóhelynél kézzel beállít a fuvarszervező egy egyszeri szállítási alkalmat a megrendelő kérése szerint.

A jóváhagyott és a kézzel rögzített szállítások listáját összefűzve alakul ki az adott napra a fuvarlista, járatokra lebontva, amelyet a gépkocsivezetőknek ki lehet adni.

2.2.4.5 Telefonkönyv és címjegyzék egység

A partnerekkel, illetve telephelyek vezetőivel való kapcsolattartás könnyítésében játszik szerepet. Itt elérhető az összes partner, illetve a felrakóhelyeinek telefonszámai, címei, email címei, esetleges egyéb kapcsolattartási adatok. A könnyebb áttekintés miatt külön van kezelve a partner és a felrakóhely telefonkönyve illetve címjegyzéke.

Az egység célja: gyors alkalmankénti keresés, teljes partner lista illetve járatok szerinti lista készítése. Ez utóbbi a gépkocsivezetők számára jelent könnyebbséget, mivel ha bármi probléma van, a listája alapján el tudja érni a telep vezetőjét, ezzel gyorsabb ügyintézést lehetővé téve.

2.2.4.6 Adminisztrációs egység

Ennek az egységnek kétféle feladata van, egyrészről itt lehet felhasználókat létrehozni, karbantartani, illetve jogokat kiosztani nekik az egyes működési egységekre. Másrészt a későbbi rendszerparamétereket is itt lehet majd beállítani (például az egyes szolgáltatások árait, egyes cikkszámokhoz tartozó feldolgozási kód, stb.)

2.2.5 A diplomamunkában megvalósításra kerülő rendszer

A diplomamunka az eredeti rendszertől független, különálló rendszert foglal magában, mely az anyag- és fuvarszervező munkakörben dolgozók feladatait könnyíti meg, teszi áttekinthetőbbé. A diplomamunkában elkészítendő rendszerben számlázás nem történik, ez továbbra is az eredeti rendszer feladata lesz.

Diplomamunkám keretein belül a fent felsorolt modulok közül a program mérete és bonyolultsága miatt csak az alábbiak kerülnek megvalósításra:

- Partner egység
- Felrakóhely egység
- Szerződés egység (csak a partnerhez tartozó részegység)
- Adminisztrációs egység (csak a felhasználó, csoport és jogkezelés)

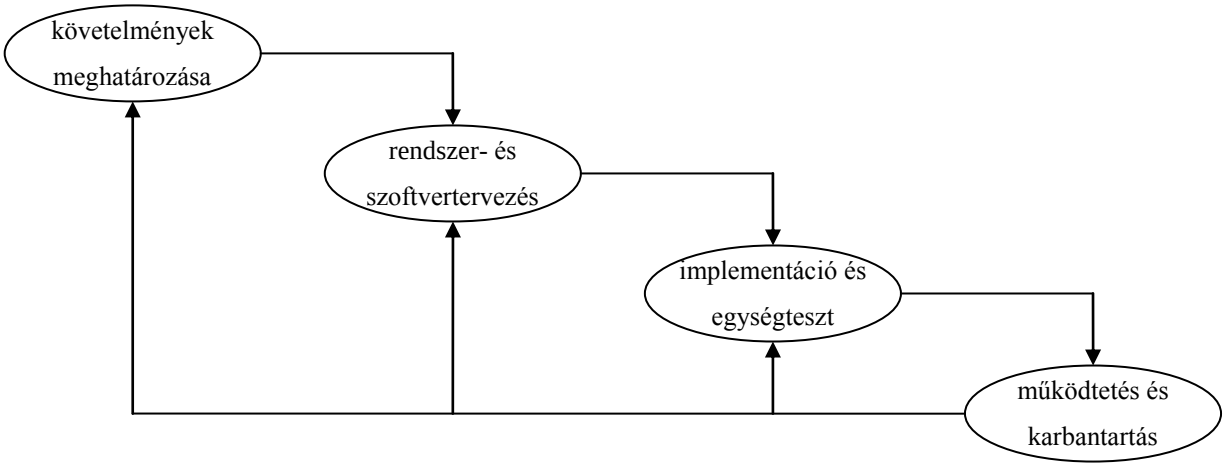
3 Fejlesztési folyamatmodellek

3.1 Általánosságban

Az elmúlt néhány évtizedben a szoftverfejlesztés kényes folyamatára szabványos folyamatmodellek alakultak ki. Minden egyes ilyen folyamatmodellnek vannak előnyei és hátrányai, megvan az alkalmazási területe, mikor célszerű és mikor nem célszerű használni. Az elmúlt 30 év folyamán a folyamatmodellek folyamatosan alakultak ki, ugyanakkor nem lehet egy adott folyamatmodellt elavultnak nevezni, ezek mindegyike él ma is. Még a tervezés előtt el kell dönteni, hogy az adott szoftver fejlesztéséhez melyik modellt követjük. Természetesen lehetőség van a modellek keverésére is, ha egy részfolyamatot egy másik modell jobban jellemez.

3.2 Vízesés modell

Az első folyamatmodell, amely kialakult, 1970-ben definiálták. Ez más termékelőállítási folyamatmodellből származik.



A vízesésmodell a nevét onnan kapta, hogy a folyamatelemek, az egyes lépések meglehetősen mereven egymásra épülnek, lépcsőzetesen kapcsolódnak egymáshoz, mint ahogy a víz folyik le a sziklán. Ez a modell előnyeit és hátrányait egyaránt magába foglalja.

Hátránya: merev, nem flexibilis egymásra épülés.

Minden egyes folyamat teljes körűen lezárul, mielőtt a következő folyamat elindulna. Ezt a felső nyilak jelzik. Ez a merevség oda vezet, hogy egy implementációs tervnek teljesen készen kell lennie. Minden egyes lépés végén születik egy dokumentum: az első és második lépés esetén ténylegesen szöveges, rajzos dokumentum, harmadik lépés után egy szoftver kód a hozzátartozó dokumentumokkal. Ha ezt a dokumentumot a felhasználó és az informatikus együtt elfogadja, akkor lehet áttérni a negyedik lépésre. Az egyes lépésekről később, külön-külön, amikor odaérünk.

Ez is, mint minden elvi modell, absztrakt modell, azonban a gyakorlatban az egyes lépések átfedik egymást, de mindenféleképpen igaz, hogy ebben a modellben nincs párhuzamos fejlesztés, a lépések semmiképp sem párhuzamosak.

Előnye kicsi és közepes méretű rendszereknél jelentkezik, és azon rendszereknél, amelyeknél a követelmények valóban jól meghatározhatók. Ekkor egy jól strukturált, jól felépített, jó architekturális jellemzőkkel rendelkező, robusztus rendszer fejleszthető.

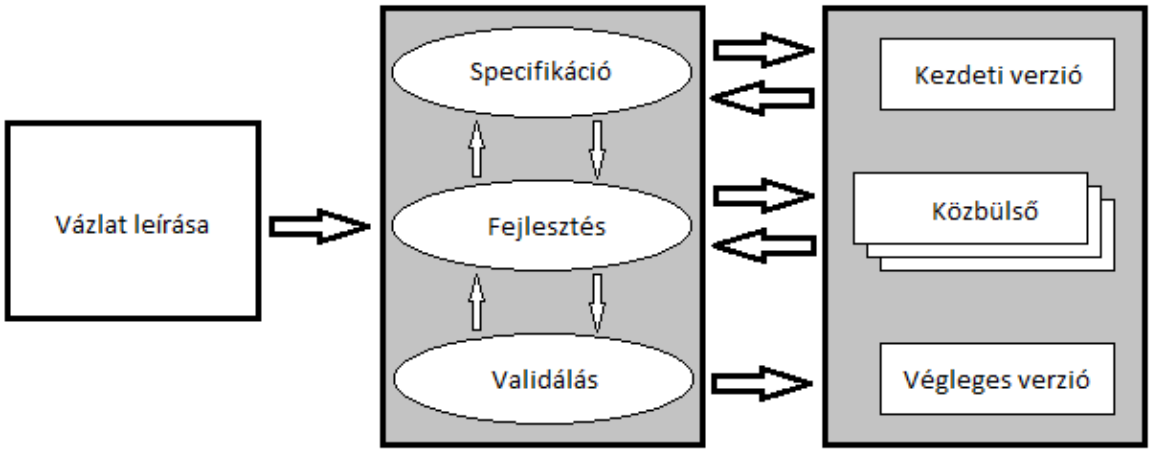
Sajnos azonban az esetek nagy részében nem igaz, hogy az elején a követelmények jól kideríthetők és rögzíthetők. Itt kezdődnek a gondok, mivel a lépések közül a működtetés a leghosszabb, és sajnálatos módon működtetés közben derülnek ki a követelményhibák, a rendszerterv hibái. Ekkor az egészet újból kell kezdeni. A legutolsó lépésben derülnek ki a korai lépések problémái: nem teljesített követelmények, félreértések, hibás specifikációk, félreérthető terv stb.

Az alsó nyilak azt jelzik, hogy bárhonnan újrakezdődhet a folyamat, de az összes lépést végig kell csinálni. Nyilvánvalóan, ha a vízesés modellben fejlesztünk rendszert, melynek világos a struktúrája, de nem ér semmit, újra kell kezdeni az egészet, ez újbóli plusz költséget és plusz időt jelent.

Viszont a vízesés modell létezik, és a vízesés modellnek mindenféle változata jött létre. Bizonyos esetekben, napjainkban is a vízesés modellt használják, mivel a legegyszerűbb, a legjobban alkalmazható modell a fenti feltételek fennállása esetén.

3.3 Evolúciós modell

Az evolúciós modellt a 80-as évek elején alkották meg, nagy rendszerek fejlesztésére. Egyik alapötlete, hogy az előbbi abszolút lineáris vagy minimálisan átfedő modellel szemben megpróbál egy párhuzamos fejlesztési modellt kialakítani. A másik ötlet, hogy a fejlesztést úgy végezzük el, hogy egy nagyon korai implementációt készítünk, majd implementációk verzióin keresztül jutunk el a végső verzióhoz. Vagyis az evolúciós fejlesztés verziósorozatot produkál.



Tehát van egy vázlatos követelményrendszer, utána párhuzamosan követelményspecifikáció, implementálás, validálás történik úgy, hogy van egy kezdeti implementáció, majd annak vannak valamilyen változatai és végül megszületik a végleges verzió. Ezeket a verziókat hívja a modell prototípusoknak. Tehát ez a prototípusorientált vagy prototípusalapú modell. A párhuzamosság és a verziókezelés nagyon gyors visszacsatolási lehetőséget jelent, gyors beavatkozást tesz lehetővé. Tehát nem a végén derülnek ki a hiányosságok, hanem már menet közben fény derül a problémákra.

Két fajta evolúciós megközelítés létezik:

- *feltáró fejlesztés:* A prototípusok segítségével a felhasználó és a fejlesztő közösen finomítja, pontosítja a követelményeket, vagyis prototípus sorozaton keresztül tárjuk fel az igényeket. Tehát a felhasználó kap egy működő verziót, ezek alapján pontosítja a követelményeket, a további fejlesztésnél ezt figyelembe vesszük és a következő verzió már jobban teljesíti a követelményeket. Így a verziósorozat egyes állomásain egyre több felhasználói igény kerül beépítésre, míg a végső verzió már a tényleges követelményeknek megfelelő lesz. Ez lesz maga a rendszer. Ilyenkor az első prototípusoknak a rendszer azon részeivel kell kezdődnie, amelyek tiszták, világosak, amelyeknek a követelményei egyértelműek, a folyamat a rendszer ismert részeinek kifejlesztésével kezdődik. A kevésbé ismert, zűrösebb rendszerelemek a későbbi prototípusokba kerülnek bele.

Minden prototípus az előzőből következik, az előző prototípusra épül, a végleges változat prototípus sorozat alapján készül el.

- *eldobható prototípus:* Adunk a felhasználó kezébe egy verziót, és ennek segítségével a felhasználó kipróbálja a prototípust, így rájöhet, hogy mit akar. Ha arra jön rá, hogy nem ezt akarja, a prototípust eldobjuk, újrafogalmazzuk a követelményrendszert, és az ennek megfelelő prototípust állítjuk elő. Ebben a megközelítésben az első prototípusok a legkevésbé megértett, legkevésbé feltárt rendszerelemekkel kezdődnek azért, hogy a felhasználó kitalálhassa, mit is szeretne.

Az evolúciós modell előnyei:

párhuzamosság; a felhasználó igényeinek jobban megfelel, gyorsabban kap egy prototípust, tud vele foglalkozni, rájön, hogy a követelmények jók vagy nem. Nagyon hamar kap kipróbálható verziókat, amelyek folyamatosan finomodnak. Így nem teljes alrendszert kell kipróbálni, csak bizonyos rendszerelemeket.

Hátrányai:

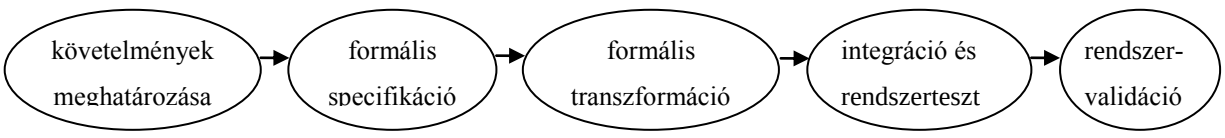
Az adott esetben meglévő túl **sok verzió** között elveszhet a felhasználó és a fejlesztő is, a folyamat egy áttekinthetetlen verziókezeléssé válik. Nagy méretű rendszerek túl sok verziója nem biztos hogy szerencsés és az említett előnyöket tükrözik.

Az evolúciós modell alapján létrehozott rendszerek – szemben a vízesés modellben létrehozott rendszerekkel – **nem robusztusak**. Magyarul a prototípus sorozat általában rossz strukturáltságú, architektúrájú szoftverhez vezet.

A prototípus-alapú fejlesztés **speciális**. Ez gyors fejlesztést, rapid technikákat igényel, amelyhez speciális technikák és speciális tudású emberek kellenek.

3.4 Formális fejlesztési modell

A formális modell a 70-es években alakult ki a vízesés modell egy változataként.



A követelményspecifikáció és a rendszervalidáció megegyezik, az e két folyamat között levő életszakasz más. Ezen formális modell szerint ugyanis a rendszerspecifikációból matematikai eszközökkel formálisan generálunk működő programot. Ehhez az szükséges, hogy a rendszerspecifikációt is formálisan, absztrakt eszközökkel adjuk meg. Ekkor a lényeg a transzformációs folyamat, amely a formálisan megadott követelményekből előállít egy adott nyelvű kódot.

Megtakarítjuk azt a lépést, amelyet mindenhol majd validálásnak nevezünk, mivel az egész modell a validáláson alapszik. Ehelyett van egy matematikai eszközrendszer, amellyel ezt elvégezzük.

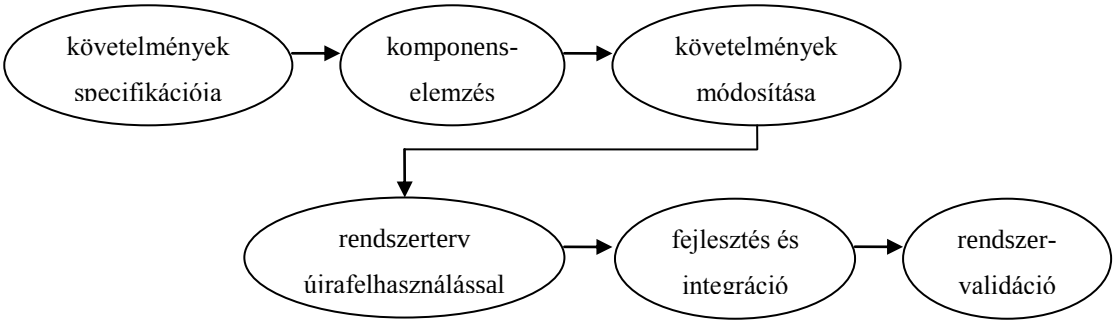
A modell Mills-től származik. Egy konkrét alkalmazása a 80-as évek második felében az IBM-Cleanroom folyamat.

Előnyei: automatizált a folyamat; bármilyen rendszer esetén alkalmazható. Egyetlen apró hibája, hogy a rendszerkövetelmények formalizálását kézzel kell elvégezni, a rendszerkövetelmények matematikai eszközökkel való felírása pedig már általában nem megy.

Biztonságkritikus rendszerek esetén alkalmazták, alkalmazzák ezt a modellt.

3.5 Újrafelhasználás alapú modell

A modell lényege a nevében szerepel, azonban az újrafelhasználás nem csak kód újrafelhasználást jelent, hanem tervezési szinten való újrafelhasználást is. A modell azon alapul, hogy léteznek olyan komponensek, amelyeket újrafelhasználhatunk. Legyen a modell olyan, hogy ezeket a meglévő komponenseket valóban újra felhasználjuk. Ez viszont a következőképpen módosítja a folyamatot:



Ha megvan a követelmények specifikációja, akkor elsőként megnézzük, hogy a világban vannak-e olyan szoftverek, netalán olyan tervelemek, amelyek nekünk jók, felhasználhatók a mi rendszerünkben; megkeressük ezeket az elemeket.

Problémák:

- tudnunk kellene, hogy vannak ilyen komponensek
- tudnunk kellene, hogy egyáltalán milyen komponensek vannak
- meg kellene tudni találni ezeket a komponenseket
- ha meg is találtuk, ezek nagy valószínűséggel nem felelnek meg teljes mértékben az igényeinknek.

Megoldásként módosítjuk a követelményeinket annak megfelelően, hogy a kész komponenseket be tudjuk építeni a rendszerünkbe. Azután ennek megfelelően végrehajtjuk a tervezést és beépítjük a tervbe vagy az implementációba a komponenseket.

Hátrány: valószínűleg kompromisszumot kell kötnünk.
Előnyök: kész komponenseket használunk fel, ezzel időt, pénzt takarítunk meg; a komponensek megvannak, léteznek, kipróbáltak, teszteltek stb., feltételezhetjük, hogy nem hibásak, vagy legalábbis minimálisan hibásak, így egyfajta biztonságérzetet nyújt, sok validációs lépést megspórolhatunk.

Probléma még, hogy leromolhat a rendszer struktúrája. Az újrafelhasználással fejlesztett rendszer rovására mehet egy bizonyos idő után, ha túl sok kompromisszumot kötünk, még akkor is, ha a komponensek jók.

3.6 Iteratív modellek

3.6.1 Spirális fejlesztési modell

A legkésőbb, 1988-ban kialakult modell, az ún. Boehm-spirál. Ez teljesen más modell, mint az eddigiek. Középpontjába olyan dolog került, amivel az eddigi modellek nem foglalkoztak: a fejlesztés kockázati tényezőinek felmérése, a kockázatból származó problémákra való felkészülés, a sikertelen kimenetelek, a rizikó csökkentése.

Négy alaplépés van, ezeket spirálisan ismételgetjük, tehát a következő spirálban ugyanaz a lépés az eddigieknél magasabb szinten ismétlődik meg.

1. lépés: a követelmények, pontosabban célok meghatározása: az adott spirális fázisban mit fogunk megvalósítani? Azonosítjuk a tevékenységeket, meghatározzuk a megszorításokat, a menedzselési tervet és azt, hogy milyen kockázati tényezők várhatók ebben a lépésben, és ezeket a kockázatokat hogyan fogjuk kezelni, milyen alternatívák lehetnek a kezelésre.

2. lépés: kockázatelemzés, kockázatok felismerése, kockázatok tényezőinek megszűntetése, ezekre való felkészülés: megtervezzük azokat a lépéseket, amelyek azt célozzák, hogy a kockázatokat el tudjuk kerülni. (Ha például a követelmények meghatározása kritikus pont, tehát úgy gondoljuk, hogy a teljes fejlesztésben nagyon nagy kockázatot jelent, rosszak a követelmények, akkor egy feltáró evolúciós modellt alkalmazzuk ebben a lépésben, építenünk kell egy prototípus sorozatot a követelmények minél pontosabb meghatározása érdekében.)

3. lépés: fejlesztés és validálás: a 2. lépés után fejlesztési modellt választunk, adott esetben minden egyes spirálban más-más fejlesztési modell alkalmazható, adott esetben a kockázatanalízis eredményeként választható a fejlesztési modell. Ezek után hajtjuk végre a szokásos tervezés, implementáció, validálás lépéseket. Az első spirálok esetén nyilvánvalóan a követelményfeltárás, követelménytervezés és a validációs lépések következnek, amíg el nem jutunk odáig, hogy...

4. lépés: tekintsük át a teljes projektet, az eddigi fejlesztést, az eddigi spirálokat, elemezzük ezeket, és döntsünk a folytatásról olyan értelemben, hogy megállunk-e (a megvalósíthatósági esettanulmány után dönthetünk úgy, hogy befejezzük). Amennyiben a folytatásról döntünk, a projektet tervezzük meg, azt tervezzük meg, hogy hogyan folytatódjon a projekt, és kezdődik elölről a spirál.

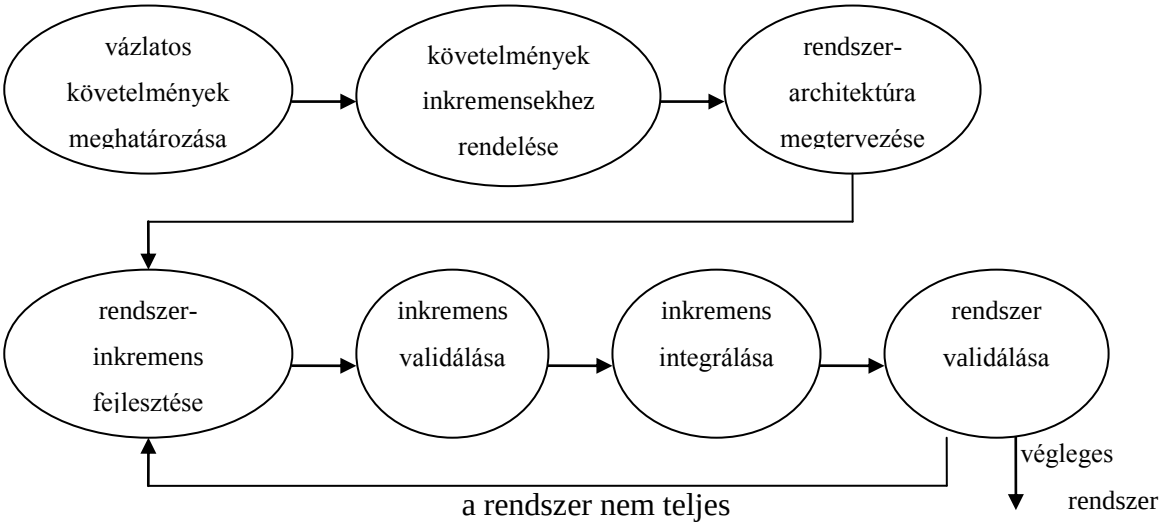
Előnyei:

- foglalkozik a kockázatokkal
- sokkal szisztematikusabban foglalkozik a projekttel

Hátránya: nem a legtriviálisabb modell.
Nagy rendszereknél javalsolt, előnyei ezeknél a rendszereknél használhatóak ki.

3.6.2 Inkrementális fejlesztési modell

A fejlesztés implementálás része kisméretű egységekben, inkremensekben történik, a mindenkori rendszerbe inkremenseket építünk, inkremensekkel bővítjük, és ha szükséges, nyilván megismételjük az inkremenshez akár a specifikációs tervezés lépését is.



Az alsó rész, az inkremensek hozzáillesztése, ciklikus folyamat, mindaddig folytatjuk, amíg úgy nem döntünk, hogy kész van a rendszer. Az inkrementális fejlesztésnek ez a ciklus a lényege. Az inkremenseket önállóan fejlesztjük, tervezzük, implementáljuk, validáljuk, sőt adott esetben még a követelményspecifikációt is megadhatjuk, megváltoztathatjuk

inkremensenként. Az architektúrát változatlanul hagyva az inkremensek fejlesztése teljes életciklussal történik.

Az inkrementális modell előnyei:

- az inkremensek kicsik (úgy kell megtervezni, hogy kicsik legyenek), ezekre önmagukban a legmegfelelőbb modell alkalmazható, például vízesésmodell is (az inkremensek kicsik, jól körülhatárolhatók, adott esetben a követelmények egyértelműen specifikálhatók).
- az inkremensek fejlesztése történhet párhuzamosan
- kezdhetjük a fejlesztést a legfontosabb funkciókat realizáló inkremensekkel, és mivel az inkremensek kicsik, a felhasználó nagyon hamar kap egy nem eldobható prototípust, amely viszont már nem a követelmény feltárásához való, hanem már tudja használni. Tehát bizonyos funkciókat, a legalapvetőbb funkciókat tartalmazó rendszert nagyon hamar megkapja a felhasználó, a kevésbé lényeges funkciókat majd később beépítjük a rendszerbe. A rendszervalidálás mindig megtörténik, tehát a rendszer ilyen értelemben a beépített inkremensekkel önmagában egy részrendszer, tehát egy használható rendszer.
- a rendszerfejlesztés kockázata kisebb, mint az összes többi modellnél (a fejlesztések igen nagy része, több mint 50 százaléka sikertelen), mivel a rendszer validált, kis elemekkel, tehát van egy validált működő rendszer még akkor is, ha befullad a projekt, legfeljebb nem teljes funkcionalitással. Nagyon nagy a valószínűsége, hogy az első pár inkremensnél még nem fogy el a pénz, az idő, az ember, nem változik meg a környezet. Ezért a részsikeres befejezés valószínűbb ennél a fejlesztési modellnél.
- a fontosabb funkciókat implementáljuk először, ennek a következtében azokat a funkciókat a felhasználók rendszeresen tesztelik, a fontosabb funkciókat jóval többször használják, mint a kevésbé fontosakat. Így már a rendszerfejlesztés közben a hibák hamarabb kiderülnek, a mindenkori részrendszerben valóban a legfontosabb funkciók a legteszteltebbek.
- Tehát egy robusztusabb rendszer fejlesztéséhez vezet ez a modell.

A modell problémáját a 2. és 3. lépésben az inkremensek megtervezése jelenti: milyen funckiókat tegyünk egy-egy inkremensbe? Mit jelent egy inkremens?

Ez a felosztás, az architektúra megtervezése az, ami nagyon problémás. A modell ezen alapszik, tehát ha az elején az architektúrát rosszul tervezzük meg, akkor a későbbiekben ez gondot okoz.

Nagyon nehéz az inkremensek behatárolása, az elején eldöntjük, hogy melyek a fontosabb funkciók: Melyik funkció, mely inkremensbe kerüljön? – az inkremensek átfedhetik egymást, nem lehet a funkciókat egymástól szétválasztani.

3.7 Az ATEV partnerinformációs rendszerének folyamatmodellje

Az alkalmazás fejlesztéséhez legjobban illő fejlesztési modell az Evolúciós modell feltáró prototípussal. Azért ezt a modellt választottam, mert ugyan a követelmények meghatározásakor már egy jól behatárolt követelményrendszer kiépítésre került, viszont folyamatosan bővülő, fejlődő alkalmazásról van szó, mely a diplomamunkámon is túlnyúlik. Emiatt a kezdeti követelményrendszer alapján elkészítem a diplomamunka keretében a főbb modulokat, egységeket és az eddig elkészült rendszert a felhasználók rendelkezésére bocsátom. A visszajelzések alapján ezután tökéletesítem a rendszert és csak utána fogok hozzá a további modulok tervezéséhez és implementálásához. Előnye még ennek a modellnek, hogy az egyes folyamatlépcsők nem annyira merevek, könnyebb menedzselni.

4 A fejlesztés lépései

4.1 Követelmények meghatározása

4.1.1 A feladat összefoglaló leírása

A rendszer feladata komplex. Egy részről segíti a partnerekkel történő kapcsolattartást; szerződéseket készít elő, tárol; nyomon követi a partnerek beszállított anyagmennyiségeit; segíti a gépkocsivezetők útvonalának megtervezését; mindezen adatok alapján fontos jelentéseket, kimutatásokat, megfigyeléseket készít. A rendszer használatához csoport szintű jogosultságkezelésre van szükség, mivel a rendszert több terület munkatársai is használhatják majd. A rendszer használatához felhasználói név és jelszó megadása szükséges. A rendszer kezdeti beállításait, paramétereit az adminisztrátor állíthatja be. Az adminisztrátor feladata a felhasználók karbantartása is. A nyersanyagszervező felviheti a partnereket, azok szerződési

adatait, majd pedig felrakóhelyeket rendelhet hozzá, illetve megadhatja az itt szükséges specifikus adatokat, mint például a szerződés mellékletének adatait, a későbbiekben pedig a behozott anyagmennyiségeket állíthatja be. A fuvarszervező a már felvitt felrakóhelyekhez állíthatja be az elszállítási alkalmakat, illetve ő szerkeszti meg a fuvar terveket is a már említett módon, illetve az ehhez kapcsolódó jelentéseket készíthet el.

4.1.2 Fogalomszótár

- **Felrakóhely:** az a hely, ahol az elhullott állatokat, maradványaikat gyűjtik, innen kerül elszállításra az ATEV telephelyeire
- **Nyersanyagszervező:** Szerződéskötésért, partnerrel való kapcsolattartásért, új ügyfelek felkutatásáért, hulladék beszállítás nyomon követéséért, reklamációk kezeléséért felelős személy
- **Fuvarszervező:** A felrakóhelyekről történő elszállítások megszervezéséért, lebonyolításáért felelős személy
- **Adminisztrátor:** A felhasználók illetve a rendszerparaméterek karbantartásáért felelős személy
- **PartnerNumber (Partnerszám):** Egyértelműen azonosítja a partnert, nem változik
- **OrderNumber (Rendelésszám):** Évente új kerül kiadásra a beszállított hulladékok elkülönítésére
- **KSHNumber (KSH szám):** Statisztikai szám
- **KUJNumber (KÜJ szám):** Környezetvédelmi Ügyfél Jelölő
- **Jelleg:** A felrakóhely besorolása abból a szempontból, hogy milyen tevékenységet végez
    - **BT:** Baromfi telep
    - **ST:** Sertés telep
    - **MT:** Marha telep
    - **JT:** Juh telep
    - **BF:** Baromfi Feldolgozó
    - **VH:** Vágóhíd
    - **GYT:** Gyepmesteri telep
    - **EP:** Egyéb partner
    - **EAT:** Egyéb állattartó telep
    - **EEU:** Egyéb élelmiszeripari üzem
    - **GYH:** Gyűjtőhely
    - **HH:** Hűtőház
    - **KER:** Kereskedelem
    - **KGY:** Konzerv gyár
    - **LAK:** Lakossági
    - **BGY:** Bőr gyár

- **CountryCode (Megye kód):** A telefonkönyv szerinti megye kódok
- **RouteCode (Járatkód):** A gépkocsik körútja által lefedett területet jelzik
    - **A:** Bihari
    - **B:** Füredi
    - **C:** Füzesgyarmati
    - **D:** Hevesi
    - **E:** Nyírségi I.
    - **F:** Nyírségi II.
    - **G:** Polgári
    - **H:** Jászsági
    - **I:** Vágóhídi I.
    - **J:** Vágóhídi II.
    - **X:** Célfuvar
    - **U:** Saját eszközzel történő beszállító
- **Weighing (Mérleglés):** A beszállított anyagmennyiség súlyát melyik fél méri le
- **WeighingPlace (Mérleglés helye):** Az a konkrét telephely ahol a mérleglést végzik
- **KTJNumber (KTJ szám):** Környezetvédelmi Terület Jelző
- **EWCNumber (EWC szám):** Hulladék veszélyességi mutatószáma
- **ProcessingCode (Feldolgozási kód):**
    - **B:** Biogáz alapanyaggyártás,
    - **F:** Égetéshez történő előkészítés,
    - **Á:** Átszállítás másik feldolgozó gyárba
- **SRM:** *meghatározott veszélyes anyagok (SRM - specified risk material):* kérődzőktől származó külön jogszabályban meghatározott a fertőző spongiform encephalopathia átvitelére transmissible spongiform encephalopathies (a továbbiakban: TSE) alkalmas anyagok
- **Classes (Osztályok):**
    - **I.:** Szarvasmarha, juh, kecske (kérődzők), kedvtelésből tartott állatok (kutya, macska), önkormányzat által begyűjtött vegyes hulladék, SRM vágási hulladék, SRM szennyvízkezelési hulladék
    - **II.:** Sertés, baromfi, nyúl, ló, öszvér, szamár, SRM mentes szennyvízkezelési hulladék, III. osztályból állatorvos által átsorolt hulladék
    - **III.:** Vágóhídi hulladékok (sertés, baromfi vegyes vágóhídi hulladék)
    - **III/B:** Szőr, SRM mentes vágási hulladék
- **StatusFlag (Státusz jelző):** Feladata az adatok érvényességének jelzése
    - **A:** Aktív
    - **I:** Inaktív

4.1.3 Tárolandó adatok

A program használatához az alábbi adatokat szükséges tárolni:

4.1.3.1 Partner adatai

Az ATEV-vel szerződésben álló beszállító partner főbb adatai.

- Partnerszám (partner egyedi azonosítója)
- Rendelésszám (évenként újuló egyedi azonosító)
- Helység (partner székhelye)
- Szerződés típus (az ATEV és partner közötti szerződés típusa)
- Cím (partner pontos címe)
- Megye kód (a partner telefonkönyvben szereplő megyekódja, a cím alapján)
- Levelezési cím (levelezési cím, ha az nem egyezne meg az előbbi címmel)
- Tulajdonos (gazdálkodó szervezet esetén a tulajdonos neve)

4.1.3.2 Felrakóhely adatai

A tényleges beszállítást igénylő telep adatai.

- Felrakóhely kód (felrakóhely egyedi azonosítója)
- Név (felrakóhely neve)
- Cím (felrakóhely címe)
- Megye kód (a partner telefonkönyvben szereplő megyekódja)
- Távolságok (a felrakóhely távolsága az egyes ATEV üzemektől)
- Anyag jellegei (a beszállított anyag(ok) jellege(i))
- Járatok kódja (a beszállítást végző járat(ok) kódja(i)
- Kihelyezett konténerek (a partnerhez kihelyezett konténerek típusai és darabja)
- Osztálybesorolás (a beszállított anyag osztályozása feldolgozás szerint)
- Mérleglés (melyik fél végzi a beszállított anyag mérleglését)
- Bejelentés alapján (az elszállítás bejelentés alapján történik-e)
- Heti alkalmak (télen, nyáron a beszállítási alkalmak száma és napjai)

4.1.3.3 Szerződés adatai

A partnerrel kötött írásos szerződés legfontosabb adatai.

- Bankszámlaszám (a partner bankszámlaszáma)

- Cégjegyzék szám (a partner cégjegyzékszáma)
- KSH szám (a partner statisztikai azonosító száma)
- KÜJ szám (a parnter környezetvédelmi ügyfél jelölő száma)
- Adószám (a partner adószáma, amennyiben magánszemély)
- Adó azonosító (a partner adó azonosítója, amennyiben jogi személy)
- Anyja neve (a partner édesanyjának neve (magánszemély esetén))
- Személyi igazolvány szám (a partner szem. ig. száma, amennyiben magánszemély)
- Település azonosító (a partner statisztikai besoroláshoz szükséges azonosító)
- MVH szám (a partner mezőgazdasági és vidékfejlesztési hivatal által kiosztott azonosító száma)
- Cím (a partner szerződés szerinti címe)

4.1.3.4 A rendszer csoportjainak adatai

- Csoport neve (a felhasználói csoport neve)
- Csoport leírása (rövid leírás a csoport jogköréről)
- Aktív (aktív-e az adott csoport, csak aktív csoportoknak van jogosultságuk)

4.1.3.5 A rendszer felhasználóinak adatai

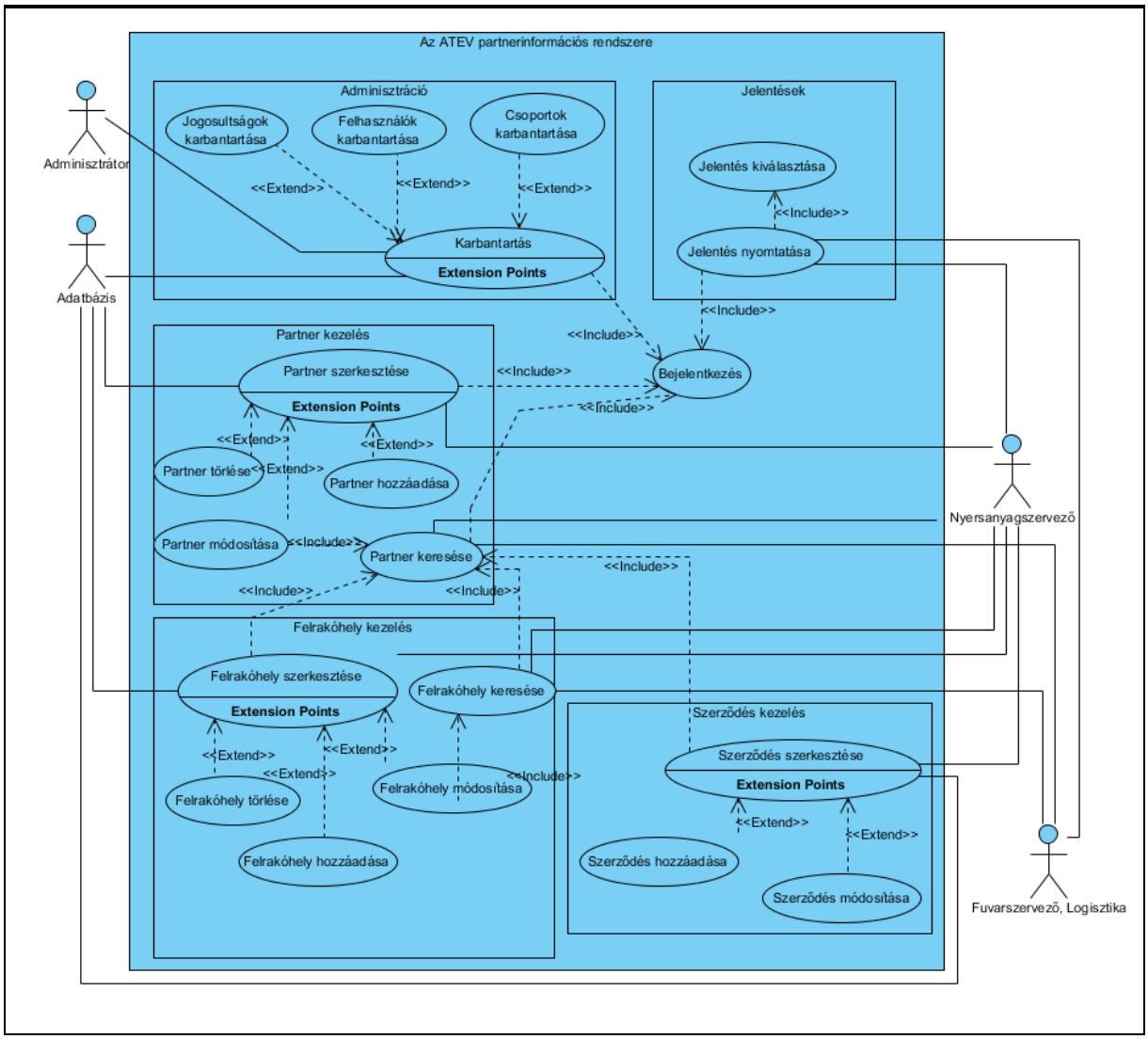
- Felhasználói név (a felhasználó neve)
- Jelszó (a felhasználó jelszava titkosítva)
- Teljes név (a felhasználó teljes neve)
- Csoport (a felhasználó csoportja, amelyikhez tartozik)
- Aktív (aktív-e az adott csoport, csak aktív csoportoknak van jogosultságuk)

4.1.3.6 Csoportok jogosultsága programmodulokhoz

- Csoport (a csoport azonosítója)
- Programmodulok (a csoport által hozzáféhető programmodulok)

4.1.4 Használati esetek

4.1.4.1 A teljes rendszer terve



4.1.5 A rendszer funkciói (Forgatókönyvek)

4.1.5.1 Adminisztrátori funkciók

Ide tartozik a felhasználók, csoportok karbantartása, illetve a jogosultságkezelés

- Felhasználók hozzáadása, szerkesztése, inaktívvá tétele, csoportba sorolása
- csoportok hozzáadása, szerkesztése, inaktívvá tétele, jogosultságok beállítása

4.1.5.2 Partnerkezelés

Általános partneradatok kezelése

- partner keresése
- partner adatainak megtekintése, módosítása, új partner hozzáadása, partner törlése

4.1.5.3 Felrakóhely kezelés

A partnerhez tartozó felrakóhelyek adatainak kezelése

- felrakóhely keresése
- felrakóhely adatainak megtekintése, módosítása, új felrakóhely hozzáadása, felrakóhely törlése

4.1.5.4 Szerződéskezelés

A partner szerződés adatainak kezelése

- szerződés megtekintése, hozzáadása, szerkesztése

4.1.5.5 Jelentéskészítés

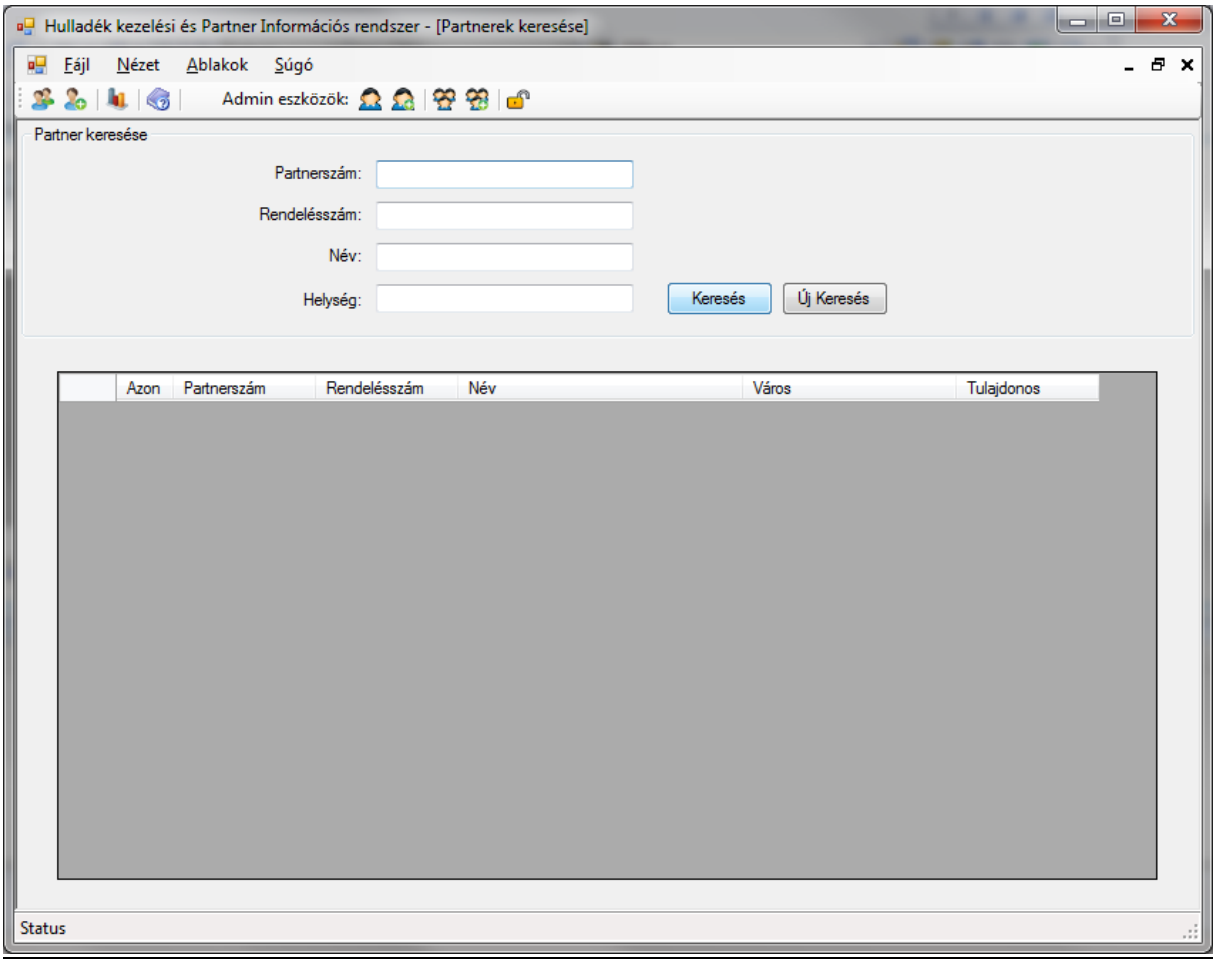
Jelentések elkészítése

- jelentés kiválasztása, jelentés paraméterezése, jelentés megtekintése, jelentés nyomtatása/exportálása

4.1.5.6 Néhány konkrét példa

Partner keresése

- kiinduló feltétel: bejelentkezett, aktív felhasználó, akinek van jogosultsága partnereket megtekinteni
- működés: a felhasználó a partnerek ikonra kattintva megnyitja a keresőablakot, megadja a keresési feltétel(eke)t, majd rányom a ’Keresés’ gombra. A megjelenő találatok közül a keresett partnerre duplán kattintva az adott partner adatlapja megnyílik. Lehetőség van az összes partner listázására is, ebben az esetben a keresési mezőket üresen kell hagyni és a ’Keresés’ gombra kattintani.

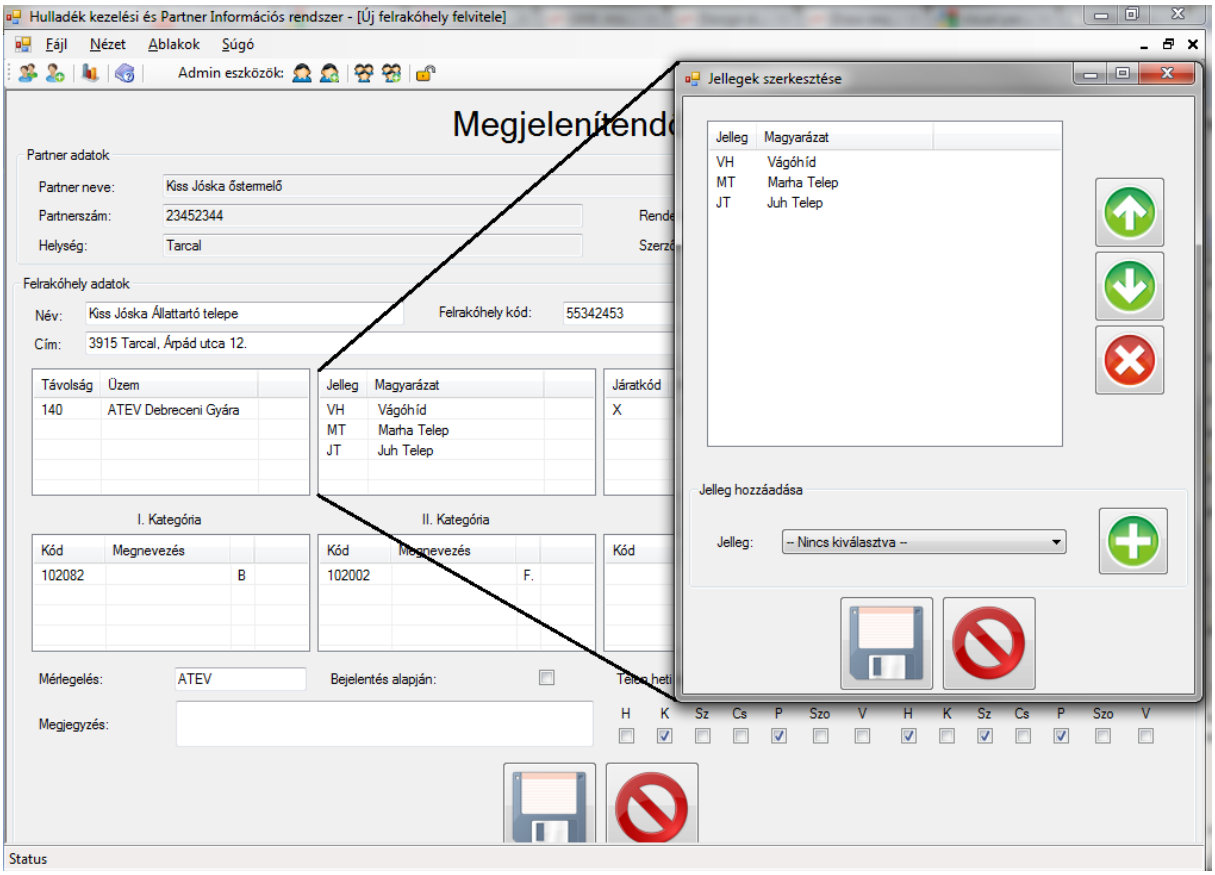


Felrakóhely hozzáadása

- kiinduló feltétel: bejelentkezett, aktív felhasználó, akinek van jogosultsága partnereket megtekinteni, illetve felrakóhelyeket hozzáadni/módosítani
- normális működés: a felhasználó először megkeresi a partnert, akihez felrakóhelyet kíván hozzáadni, majd a partner adatlapján a ’Felrakóhely hozzáadása’ gombra kattint. A megjelenő felületen kitölti az összes szükséges mezőt, illetve feltölti a listákat (a listafeltöltésre a lista fejlécére kattintva, a megnyíló ablakban van lehetőség). A listákon belül prioritások vannak. Ezt a fel-le nyíllal tudja a felhasználó a kiválasztott elemen beállítani. Miután minden szükséges mezőt kitöltött, a mentés gombra kattintva hagyhatja jóvá a mentési folyamatot.
- hibás működés: a felhasználó, nem töltött ki minden kötelező mezőt, vagy nem megfelelő formátumú karaktert adott meg (pl: heti szállítási alkalmak csak számok lehetnek). Ekkor a rendszer figyelmezteti a felhasználót, hogy a mentés

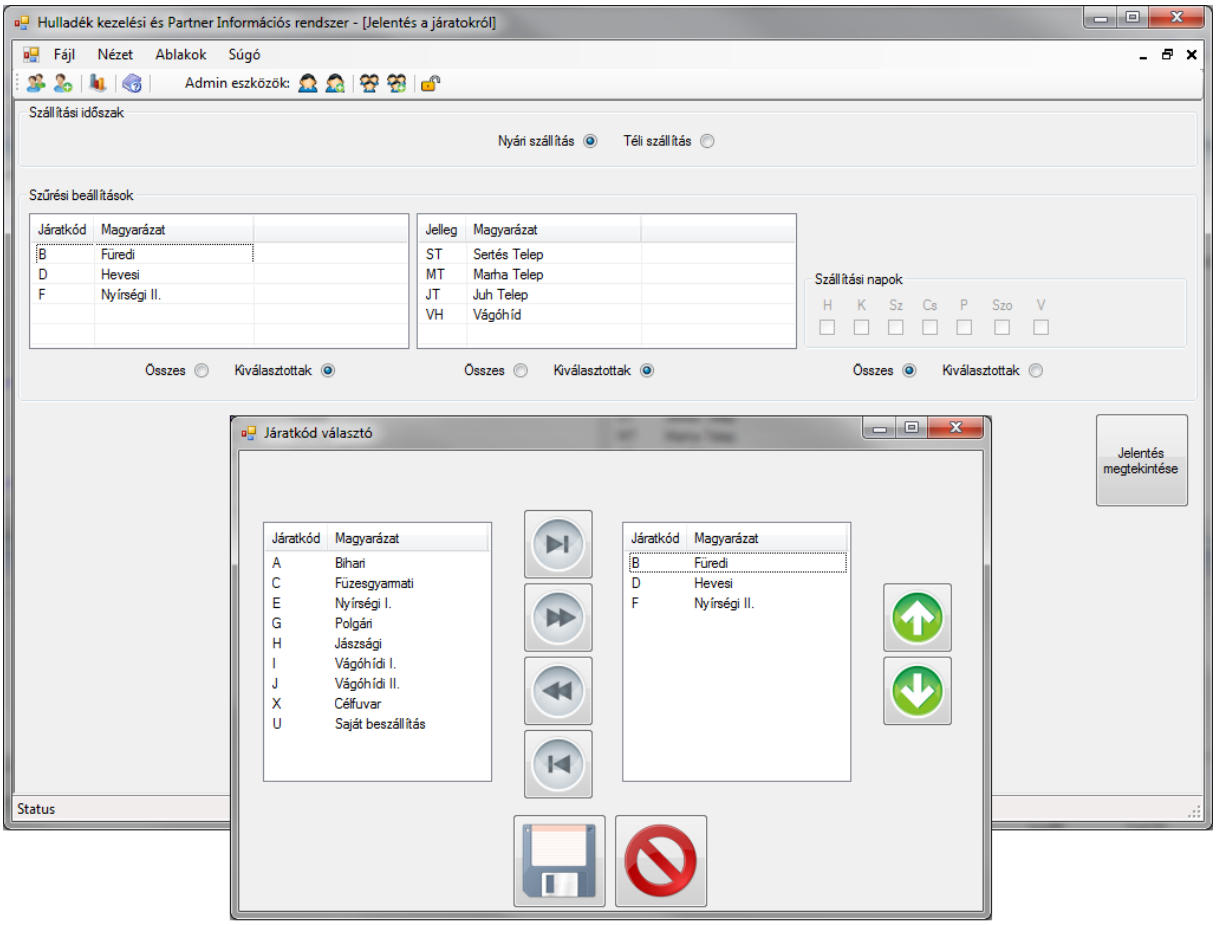
nem végezhető el, amíg a hiányosságokat ki nem javítja. A problémás mezőket a rendszer jelzi.

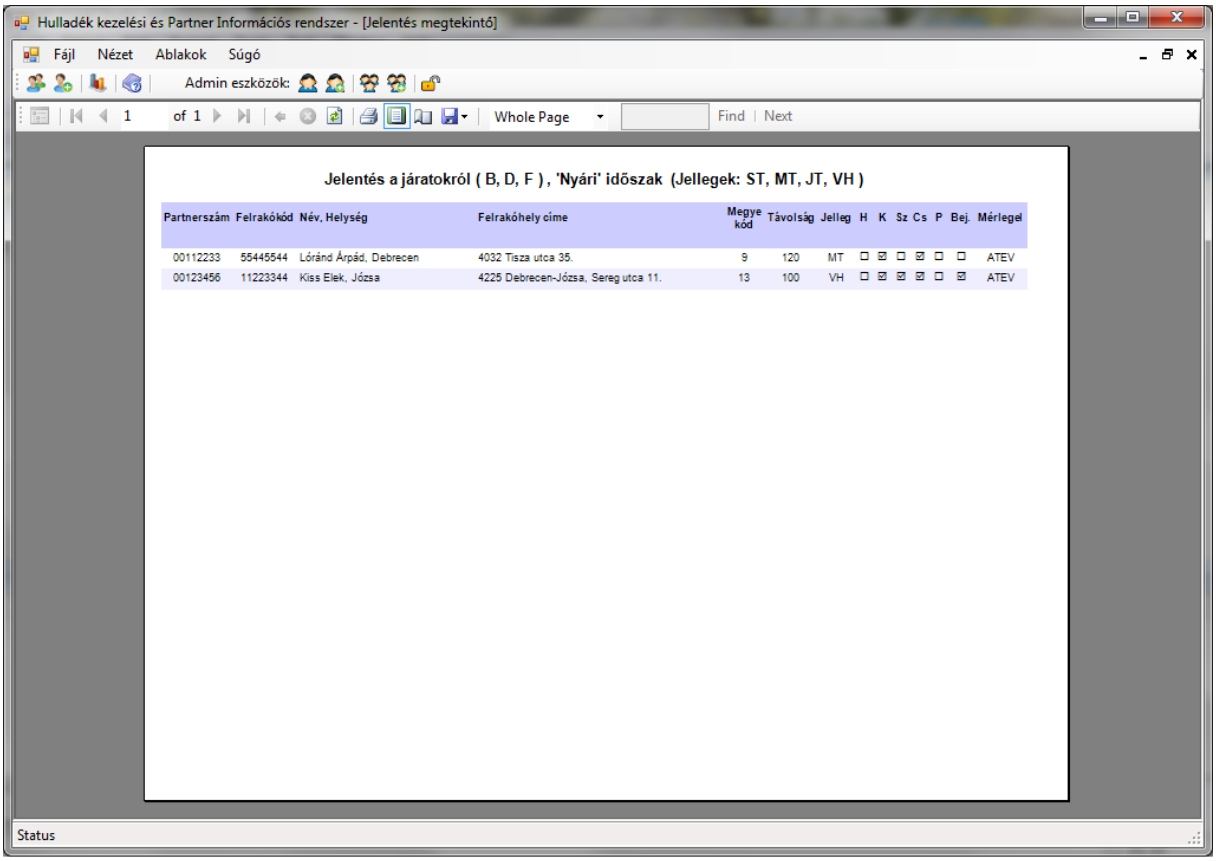
- rendszerállapot: sikeres validáció után a rendszer elmenti az új felrakóhelyet a megadott partnerhez, majd megnyitja az új felrakóhely adatlapját



Jelentés nyomtatása

- kiinduló feltétel: bejelentkezett, aktív felhasználó, akinek van jogosultsága jelentést készíteni
- működés: a felhasználó a főablakon rákattint a ’Jelentések’ ikonra. A megjelenő listából kiválasztja a számára szükséges jelentést, majd duplán rákattint, vagy az ablak alján található ’Jelentés megnyitása’ gombra kattint. Egy újabb ablakon az adott jelentést a felhasználó saját igényei szerint testre szabhatja. Ha elkészült a beállításokkal, akkor a ’Jelentés megtekintése’ gomra kattintva a rendszer betölti a megadott paramétereknek megfelelő jelentést, mely ezután nyomtatható és exportálható pdf és excel formátumba.


Hulladék kezelési és Partner Információs rendszer - [Jelentés a járatokról]
Fájl
Nézet
Ablakok
Súgó
Admin eszközök:
Szállítási időszak
Nyári szállítás
Téli szállítás
Szűrési beállítások
Járatkód
Magyarázat
B
Füredi
D
Hevesi
F
Nyírségi II.
Jelleg
Magyarázat
ST
Sertés Telep
MT
Marha Telep
JT
Juh Telep
VH
Vágóhíd
Szállítási napok
H
K
Sz
Cs
P
Szo
V
Összes
Kiválasztottak
Összes
Kiválasztottak
Összes
Kiválasztottak
Jelentés megtekintése
Járatkód választó
Járatkód
Magyarázat
A
Bihari
C
Füzesgyarmati
E
Nyírségi I.
G
Polgári
H
Jászsági
I
Vágóhídi I.
J
Vágóhídi II.
X
Célfuvar
U
Saját beszállítás
Járatkód
Magyarázat
B
Füredi
D
Hevesi
F
Nyírségi II.
Status



4.1.6 Adatvédelem és biztonság

Az alkalmazás csak bejelentkezés után használható, így illetéktelenek nem férhetnek hozzá érzékeny adatokhoz. Ezen felül minden felhasználó tagja egy csoportnak, amely csoportnak jól behatárolt jogosultságai vannak az egyes programmodulok felett. Minden programmodult csak a hozzá való jogosultsággal lehet használni, egyáltalán elérni, ugyanis az alkalmazás jogosultság hiányában fel sem ajánlja az adott modul megnyitását. Minden modul megnyitását a Fő Form végzi, mely ellenőrzi minden ablak megnyitásakor, hogy az adott ablakra van-e a bejelentkezett felhasználónak jogosultsága, amennyiben nincs, ignorálja az ablak megnyitását és hibaüzenetet jelenít meg.

Az alkalmazásban tetszőleges számú felhasználó és csoport hozható létre, viszont egy felhasználó csak egy csoport tagja lehet. A jogosultságok tetszőlegesen testre szabhatók minden csoport részére.

Lehetőség van a felhasználók, illetve csoportok inaktívvá tételére is. Ebben az esetben az adott felhasználók és az adott csoportok felhasználói a továbbiakban nem fogják tudni elérni a rendszert.

4.1.7 Felhasználói felületek

A program felhasználói felületei a Visual Studio 2008-as integrált fejlesztői környezet segítségével készültek. Ezen felülettervek részét képezik a már készülő alkalmazásnak. Terjedelmi okokból ehhez a ponthoz nem csatolok képet, de néhány felületterv megtekinthető a forgatókönyveknél, a konkrét példáknál. A teljes képernyő kép lista a CD mellékleten megtalálható.

4.2 Tervezés

A fejlesztés első nagy lépcsője ezzel lezárult, feltártuk az alkalmazás tervezéséhez és implementálásához szükséges kiindulási követelményeket és rögzítettük ezeket a megfelelő dokumentumokban. Ezek a dokumentumok nagyon fontosak, mivel ezek alapján készíti el a fejlesztő az alkalmazásunkat. Azért, hogy mind a mezei felhasználó, mind a fejlesztő megértse, ezeket a dokumentumokat valamilyen logikai modell, vagy pedig egy informális jelölésrendszer segítségével rögzítjük. Én munkám során az UML logikai modellt választottam. Azért esett erre a választásom, mert széles körben alkalmazott, jól áttekinthető, egyértelmű összefüggéseket ír le.

4.2.1 Tervezői eszköztár

Az UML diagramok elkészítéséhez a ’Visual Paradigm for UML’ szoftvert alkalmaztam, mivel könnyen használható, sokoldalú UML tervező alkalmazás.

Az adatbázisterv elkészítéséhez a ’Fabforce: DBDesigner Fork’ alkalmazást használtam. Ennek a szoftvernek hatalmas előnye, hogy biztosítja a tervezés folyamán mindazt az eszközrendszert, amit egy ilyen szoftvertől elvárunk és mindemellett a munka befejeztével legenerálhatjuk az elkészült terv alapján a CREATE scriptet, mellyel az adatbázisunk létrehozható. Több adatbázismotort is támogat, többek között a MySql, ORACLE és MSSQL adatbázisokat is. Egyetlen hátránya, hogy a felhasználónak ismernie kell, hogy az adott adatbázisrendszerben milyen típusú változók vannak, ugyanis a DBDesigner tervezéskor nem köti meg a kezünket a típusválasztásban, ugyanakkor egy nagyobb méretű adatbázis létrehozásakor csúnya hiba tud lenni, ha sok az eltérő adattípus.

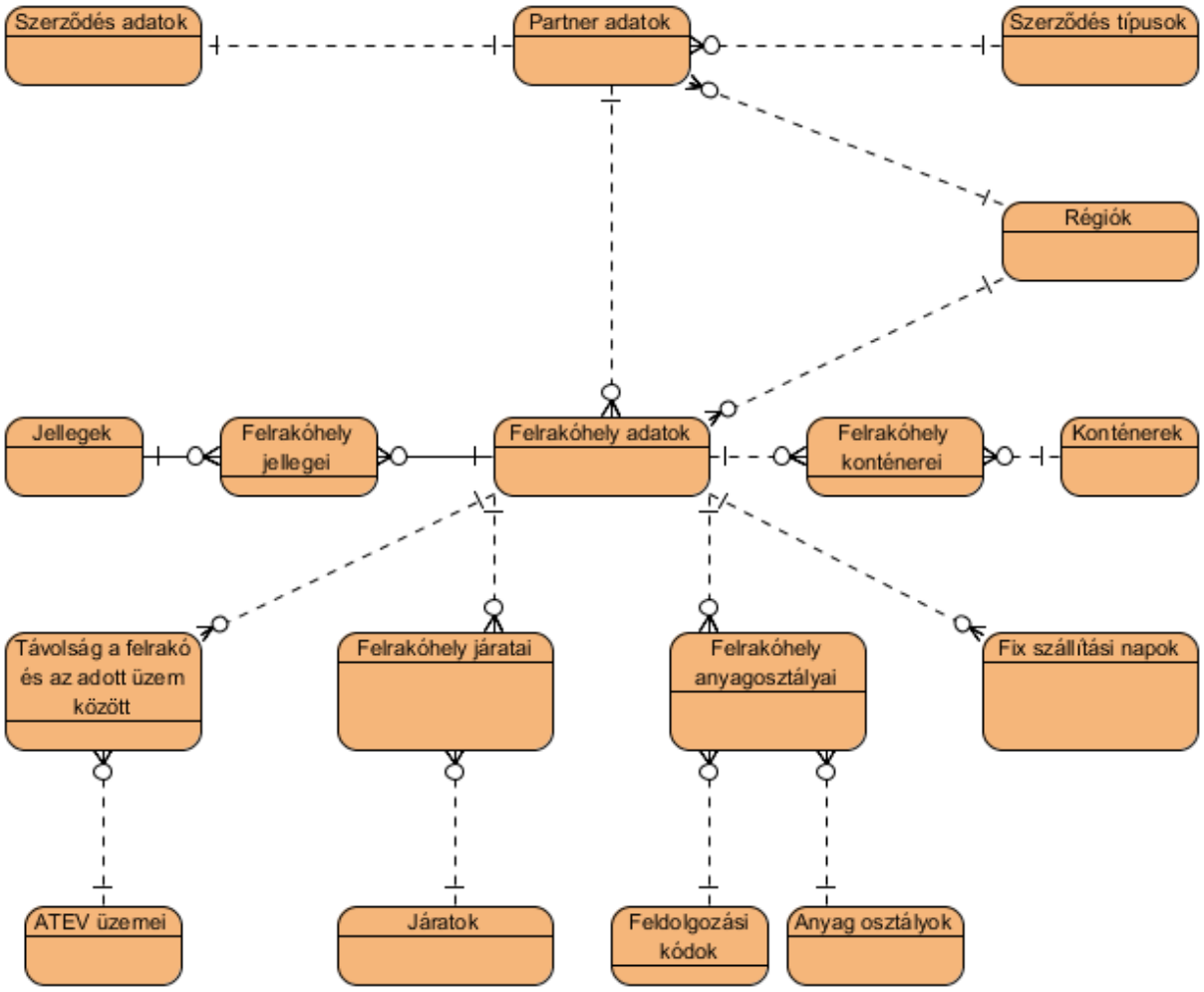
4.2.2 Adatbázis terv

A tervezés egyik legkényesebb pontja az adatbázis struktúra felépítése. Fontos megtalálnunk az egyensúlyt az adatok ésszerű tárolása és az alkalmazásból történő minél egyszerűbb adatelérés között. Nem elég ugyanis egy elvi síkon tökéletes adatbázis, azt tudni kell használni is. Fontos szempont, hogy az adatokat lehetőleg ne tároljuk redundánsan, viszont a normál formák esetében néha kompromisszumokra is szükség van az alkalmazás gördülékeny működéséhez.

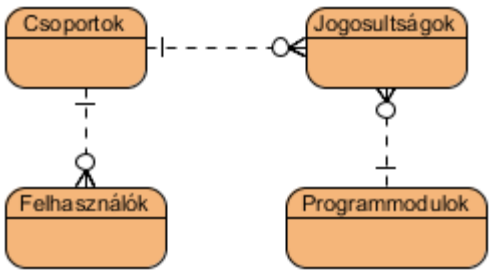
Az adatbázisban tárolt adatokat két csoportra bonthatjuk, az egyik csoport a felhasználói adatok, melyekkel az alkalmazás használói dolgoznak, illetve rendszer adatok, melyek az alkalmazás működéséhez szükségesek, például paraméterek, jogosultságkezeléshez szükséges adatok, stb. A felhasználói adatok a ’Tárolandó adatok’ fejezetben már meghatározásra kerültek.

4.2.2.1 Logikai adatterv

Az adatok közötti kapcsolatokat ER modell segítségével ábrázolom. Az alábbi ábrán a felhasználói adatok modellje látható.



A felhasználói adatokon kívül a rendszer működéséhez szükséges adatokat is tartalmazza az adatbázis. Ezen adatokat csak rendszergazdai jogosultsággal lehet manipulálni. Ezen adatok ER modellje az alábbi ábrán látható.



4.2.2.2 Fizikai adatterv

Az alkalmazáshoz elkészített adatbázis az alábbi sémaelemeket fogja tartalmazni.

4.2.2.2.1 Táblák

A táblák szerkezetét a tervezői eszközben készült képernyőképekkel szemléltetem. Az ’NN’ oszlop jelentése: ’Not Null’, az ’AI’ oszlop jelentése pedig: ’Auto Increment’

A kulcs ikon értelemszerűen az elsődleges kulcsot jelöli, míg a piros színű rombuszok a külső kulcsok. Mivel ezen külső kulcsok nem egyértelműek az ábra alapján, hogy melyik táblával állnak kapcsolatban, így ezeket külön táblázatban is feltüntettem. A kék rombusszal jelölt mezők általános mezők.

Partner adatok

Table Name: PartnerData

Column Name	DataType	NN	AI	Flags	Default Value	Comments
PartnerId	DECIMAL	✔	✔	ZEROFILL		
Contractid	DECIMAL			ZEROFILL		
ContractTypesId	DECIMAL			ZEROFILL		
RegionsId	DECIMAL			ZEROFILL		
Name	VARCHAR(255)	✔		BINARY		
Place	VARCHAR(100)			BINARY		
ZipCode	VARCHAR(50)			BINARY		
Address	VARCHAR(255)			BINARY		
PartnerNumber	VARCHAR(50)			BINARY		
OrderNumber	VARCHAR(50)			BINARY		
Administrator	VARCHAR(100)			BINARY		
LetterAddress	VARCHAR(255)			BINARY		
Owner	VARCHAR(100)			BINARY		
ContactPerson	VARCHAR(255)			BINARY		
PartnerDataComment	VARCHAR(255)			BINARY		
ValidStart	DATE					
ValidEnd	DATE					
StatusFlag	CHAR			BINARY		

Külső kulcsok:

Forrás mező	Tábla	Cél mező
ContractId	Contract	ContractId
ContractTypesId	ContractTypes	ContractTypesId
RegionsId	Regions	RegionsId

Táblamegszorítások:

UNIQUE(PartnerNumber, StatusFlag, ValidStart, ValidEnd)

Felrakóhely adatok

Table Name: DepotData

Column Name	DataType	NN	AI	Flags	Default Value	Comments
DepotId	DECIMAL	✓	✓	ZEROFILL		
PartnerId	DECIMAL	✓		ZEROFILL		
RegionsId	DECIMAL			ZEROFILL		
DepotCode	VARCHAR(50)			BINARY		
DepotName	VARCHAR(255)	✓		BINARY		
DepotPlace	VARCHAR(100)			BINARY		
DepotAddress	VARCHAR(255)			BINARY		
SiteDirectorName	VARCHAR(100)			BINARY		
SiteDircetorPhone	VARCHAR(45)			BINARY		
Weighing	VARCHAR(45)			BINARY		
WeighingPlace	VARCHAR(255)			BINARY		
isTransportByNotifica	CHAR			BINARY		
Comments	VARCHAR(255)			BINARY		
ValidStart	DATE					
ValidEnd	DATE					
StatusFlag	CHAR			BINARY		

Külső kulcsok

Forrás mező	Tábla	Cél mező
PartnerId	PartnerData	PartnerId
RegionsId	Regions	RegionsId

Táblamegszorítások:

UNIQUE(DepotCode, StatusFlag, ValidStart, ValidEnd)

Szerződés adatok

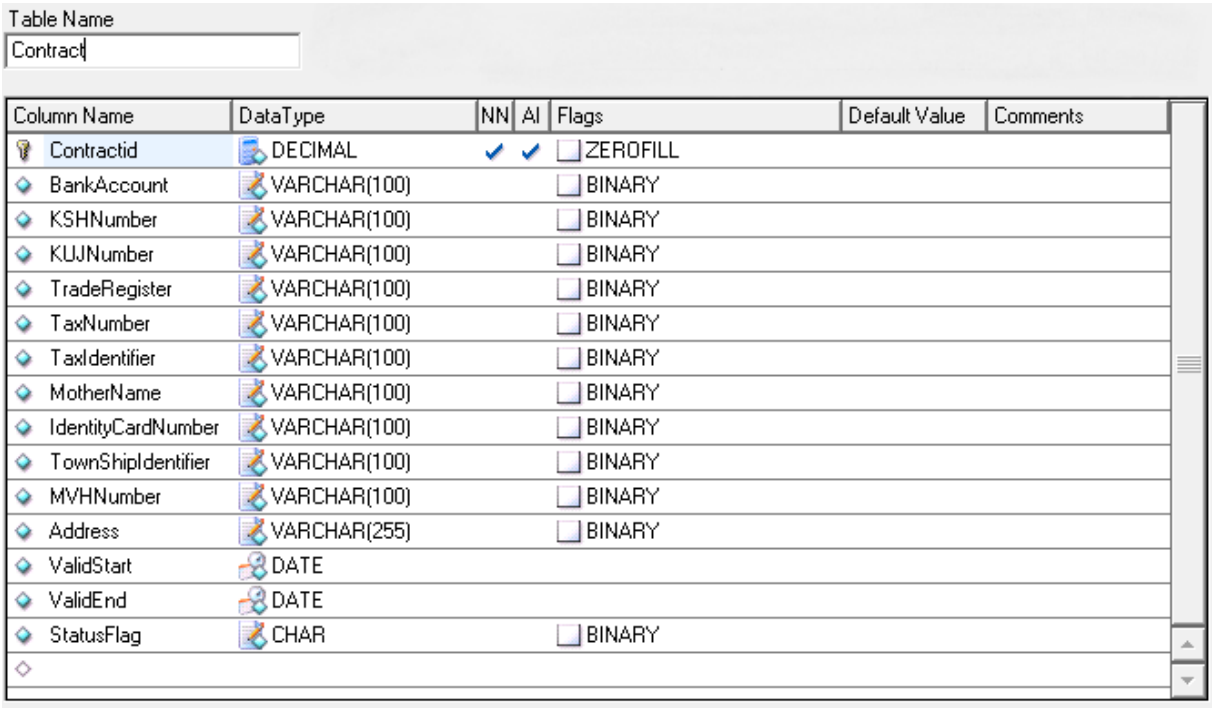


Table Name: Contract

Column Name	DataType	NN	AI	Flags	Default Value	Comments
Contractid	DECIMAL	✓	✓	ZEROFILL		
BankAccount	VARCHAR(100)			BINARY		
KSHNumber	VARCHAR(100)			BINARY		
KUJNumber	VARCHAR(100)			BINARY		
TradeRegister	VARCHAR(100)			BINARY		
TaxNumber	VARCHAR(100)			BINARY		
TaxIdentifier	VARCHAR(100)			BINARY		
MotherName	VARCHAR(100)			BINARY		
IdentityCardNumber	VARCHAR(100)			BINARY		
TownShipIdentifier	VARCHAR(100)			BINARY		
MVHNumber	VARCHAR(100)			BINARY		
Address	VARCHAR(255)			BINARY		
ValidStart	DATE					
ValidEnd	DATE					
StatusFlag	CHAR			BINARY		

Szállítási napok

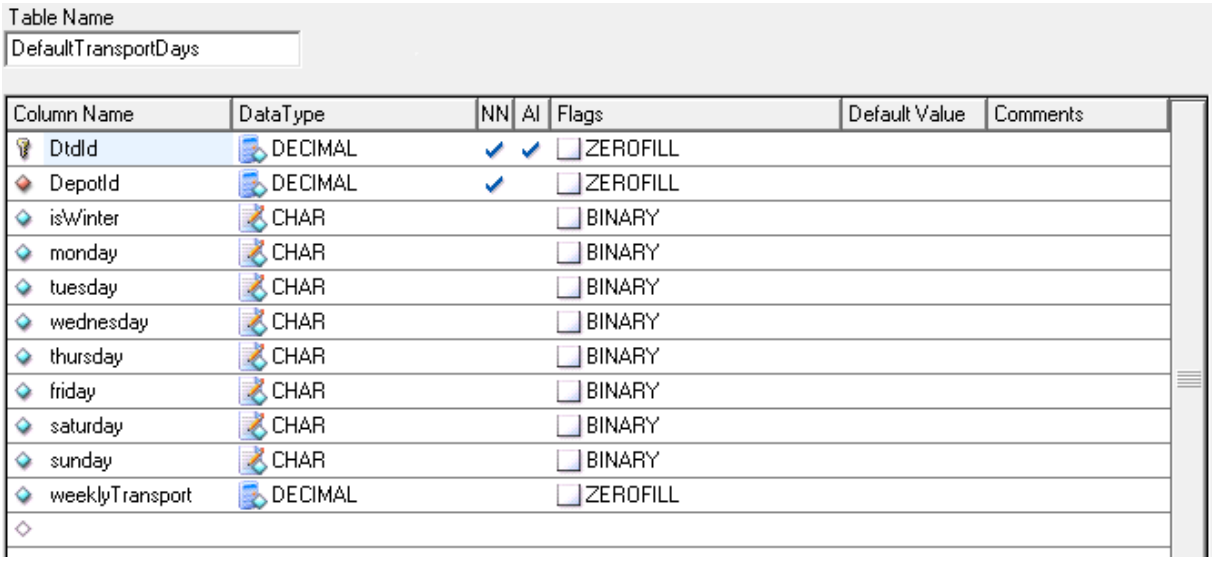


Table Name: DefaultTransportDays

Column Name	DataType	NN	AI	Flags	Default Value	Comments
DtdId	DECIMAL	✓	✓	ZEROFILL		
DepotId	DECIMAL	✓		ZEROFILL		
isWinter	CHAR			BINARY		
monday	CHAR			BINARY		
tuesday	CHAR			BINARY		
wednesday	CHAR			BINARY		
thursday	CHAR			BINARY		
friday	CHAR			BINARY		
saturday	CHAR			BINARY		
sunday	CHAR			BINARY		
weeklyTransport	DECIMAL			ZEROFILL		

Külső kulcsok

Forrás mező	Tábla	Cél mező
DepotId	DepotData	DepotId

Jellegek

Table Name: Modality

Column Name	DataType	NN	AI	Flags	Default Value	Comments
ModalityId	DECIMAL	✓	✓	ZEROFILL		
Modality	VARCHAR(100)			BINARY		
ShortModality	VARCHAR(20)	✓		BINARY		

Felrakóhely jellegei

Table Name: DepotsAndModalities

Column Name	DataType	NN	AI	Flags	Default Value	Comments
DepotsAndModalities	DECIMAL	✓	✓	ZEROFILL		
DepotId	DECIMAL	✓		ZEROFILL		
ModalityId	DECIMAL	✓		ZEROFILL		
Priority	DECIMAL			ZEROFILL		

Külső kulcsok

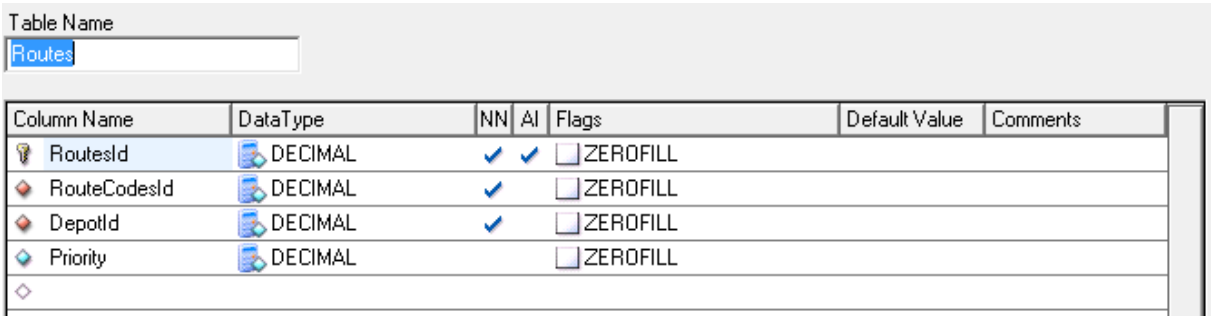
Forrás mező	Tábla	Cél mező
DepotId	DepotData	DepotId
ModalityId	Modality	ModalityId

Járatkódok

Table Name: RouteCodes

Column Name	DataType	NN	AI	Flags	Default Value	Comments
RouteCodesId	DECIMAL	✓	✓	ZEROFILL		
Code	CHAR	✓		BINARY		
Details	VARCHAR(255)			BINARY		

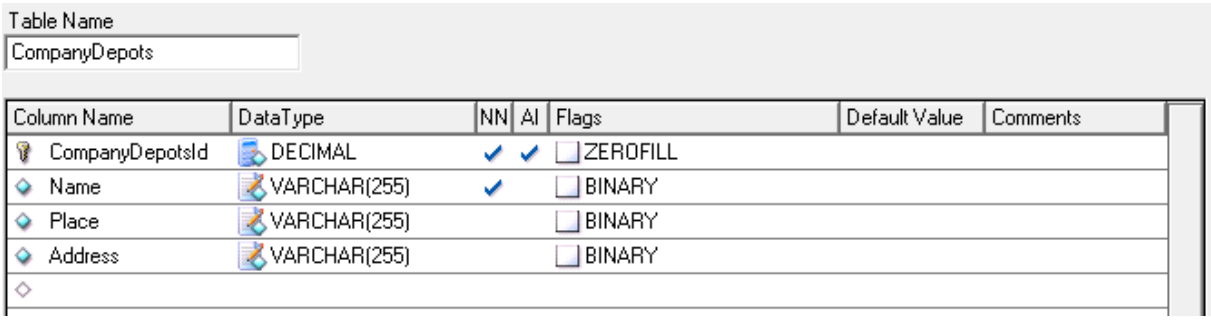
Felrakóhely járatai



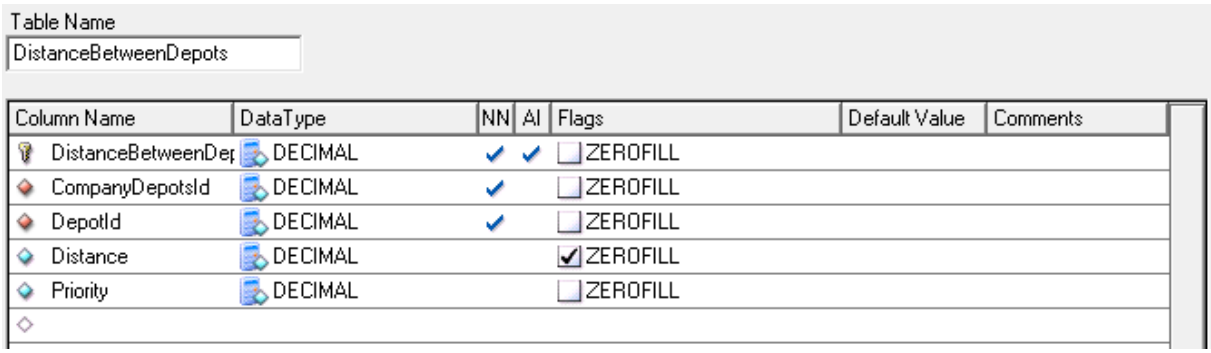
Külső kulcsok

Forrás mező	Tábla	Cél mező
DepotId	DepotData	DepotId
RouteCodesId	RouteCodes	RouteCodesId

ATEV telephelyei



Felrakóhely és az ATEV telepei közötti távolság



Külső kulcsok

Forrás mező	Tábla	Cél mező
DepotId	DepotData	DepotId
CompanyDepotsId	CompanyDepots	CompanyDepotsId

Osztályok

Table Name
Classes

Column Name	DataType	NN	AI	Flags	Default Value	Comments
ClassesId	DECIMAL	✓	✓	ZEROFILL		
ClassNumber	VARCHAR(20)	✓		BINARY		
ItemNumber	DECIMAL	✓		ZEROFILL		
ItemName	VARCHAR(255)			BINARY		
EwcNumber	DECIMAL			ZEROFILL		

Feldolgozási kódok

Table Name
ProcessingCodes

Column Name	DataType	NN	AI	Flags	Default Value	Comments
ProcessingCodesId	DECIMAL	✓	✓	ZEROFILL		
ProcessingCode	VARCHAR(5)	✓		BINARY		
ProcessingName	VARCHAR(255)	✓		BINARY		

Felrakóhely anyagosztályai

Table Name
ClassesForDepots

Column Name	DataType	NN	AI	Flags	Default Value	Comments
ClassesForDepotsId	DECIMAL	✓	✓	ZEROFILL		
DepotId	DECIMAL	✓		ZEROFILL		
ProcessingCodesId	DECIMAL	✓		ZEROFILL		
ClassesId	DECIMAL	✓		ZEROFILL		

Külső kulcsok

Forrás mező	Tábla	Cél mező
DepotId	DepotData	DepotId
ClassesId	Classes	ClassesId
ProcessingCodesId	ProcessingCodes	ProcessingCodesId

4.2.2.2.2 Nézetek

Partner adatok nézet (PartnerDataView)

Oszlopok:

- Partner szám
- Rendelés szám
- Partner neve
- Helység
- Irányítószám
- Cím
- Levelezési cím
- Tulajdonos
- Regió ID
- Régió neve
- Régió kódja
- Szerződés típus ID
- Szerződés típus neve
- Szerződés típus komment
- Szerződés ID
- Érvényesség kezdete
- Érvényesség vége
- Státusz jelző

Leírás:

A nézet az alkalmazásban levő partner adatok betöltését könnyíti meg, már adatbázis szinten előre össze vannak kapcsolva a szükséges táblák (Partner adatok, Régiók, Szerződés Típusok)

Felrakóhely adatok nézet (DepotDataView)

Oszlopok:

- Partner ID
- Felrakóhely kódja
- Felrakóhely neve
- Helység
- Felrakóhely címe
- Mérleglést végző
- Bejelentés alapján
- Komment
- Regió ID
- Régió neve
- Régió kódja
- Érvényesség kezdete
- Érvényesség vége
- Státusz jelző

Leírás:

A nézet az alkalmazásban levő Felrakóhely adatok betöltését könnyíti meg, már adatbázis szinten előre össze vannak kapcsolva a szükséges táblák (Felrakóhelyek, Régiók)

Felrakóhelyek anyagosztályai nézet (DepotClassesView)

Oszlopok:

- Felrakóhelyek anyagosztályai ID
- Osztály ID
- Feldolgozási kód ID
- Felrakóhely ID
- Osztály kategória (I, II, III, III/B)
- Cikkszám

- Cikk név
- EWC szám
- Feldolgozási kód
- Feldolgozási név

Leírás:

A nézet a felrakóhelyekhez beállított részletes anyagosztályokat és azok feldolgozási kódjait jeleníti meg.

Felrakóhelyek konténerei nézet (DepotContainersView)

Oszlopok:

- Felrakóhelyek konténerei ID
- Felrakóhely ID
- Konténer ID
- Konténerek darabszáma
- Konténer kódja
- Konténer neve
- Konténer költsége

Leírás:

A nézet a felrakóhelyekhez beállított konténereket és azok részleteit jeleníti meg.

Felrakóhelyek távolságai nézet (DistancesView)

Oszlopok:

- Felrakóhelyek távolságai ID
- ATEV üzem ID
- Felrakóhely ID
- Távolság
- Prioritás
- ATEV üzem neve
- ATEV üzem helye
- ATEV üzem címe

Leírás:

A nézet a felrakóhelyekhez beállított ATEV üzem távolságokat és azok részleteit jeleníti meg.

Felrakóhelyek jellegei nézet (ModalitiesView)

Oszlopok:

- Felrakóhelyek jellegei ID
- Felrakóhely ID
- Jelleg ID
- Prioritás
- Jelleg megnevezése
- Jelleg rövid kódja

Leírás:

A nézet a felrakóhelyekhez beállított jellegeket és azok részleteit jeleníti meg.

Felrakóhelyek járatai nézet (RoutesView)

Oszlopok:

- Felrakóhelyek járatai ID
- Felrakóhely ID
- Járatkód ID
- Prioritás
- Járat kód
- Járat részletei

Leírás:

A nézet a felrakóhelyekhez beállított járatokat és azok részleteit jeleníti meg.

Felhasználók nézet (UsersView)

Oszlopok:

- Felhasználó ID
- Csoport ID
- Felhasználói név
- Jelszó
- Teljes név
- Aktív felhasználó
- Csoport neve
- Csoport részletek
- Aktív csoport

Leírás:

A nézet a felhasználókat jeleníti meg, illetve, hogy melyik csoporthoz tartoznak és a csoport részleteit is.

Csoportok és jogosultságok nézet (GroupsAndGrantedModules)

Oszlopok:

- Csoport ID
- Program modul ID
- Modul kód
- Modul neve
- Modul részletei

Leírás:

A nézet a csoportokhoz tartozó program modul jogosultságokat jeleníti meg.

Jelentés a járatokról nézet (ReportRoutesSimpleView)

Oszlopok:

- Partner ID
- Felrakóhely ID
- Régió ID
- Partnerszám
- Felrakóhely kód
- Partner neve
- Helység
- Felrakóhely címe

- Régió kód
- Jelleg kód
- Elszállítási napok télen H-V
- Bejelentés alapján
- Távolság
- Elszállítási napok nyáron H-V
- Mérleglés

Leírás:

A nézet a járatokról történő jelentés alapját képezi.

Jelentés az állattartó telepekről nézet (ReportAnimalDepotsSimpleView)

Oszlopok:

- Partner ID
- Felrakóhely ID
- Régió ID
- Partnerszám
- Felrakóhely kód
- Partner neve
- Helység
- Felrakóhely címe
- Régió kód
- Járat kód 1
- Járat kód 2
- Távolság
- Elszállítási napok száma nyáron
- Elszállítási napok száma télen
- Bejelentés alapján
- Mérleglés

Leírás:

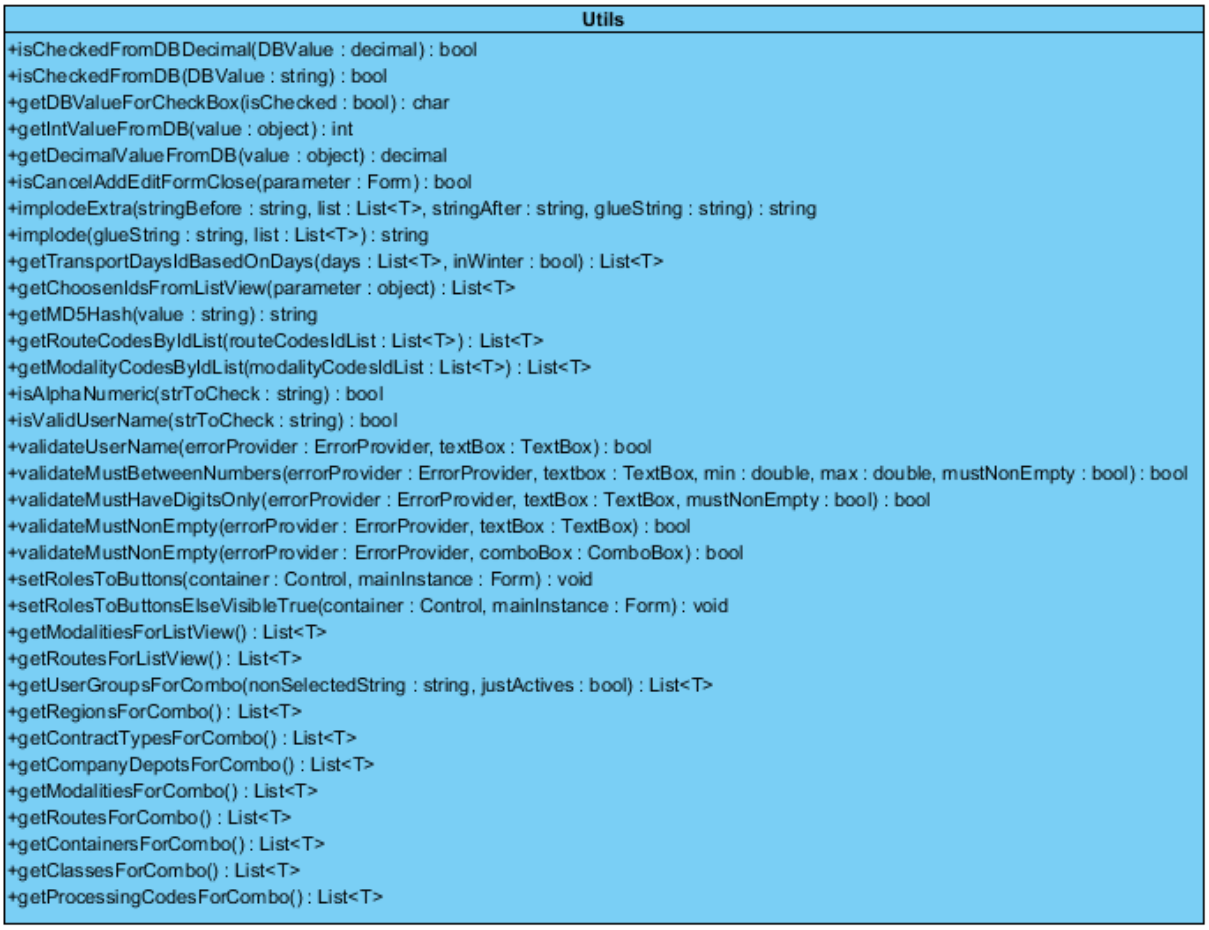
A nézet az állattartó telepekről történő jelentés alapját képezi.

4.2.3 Funkcionális terv

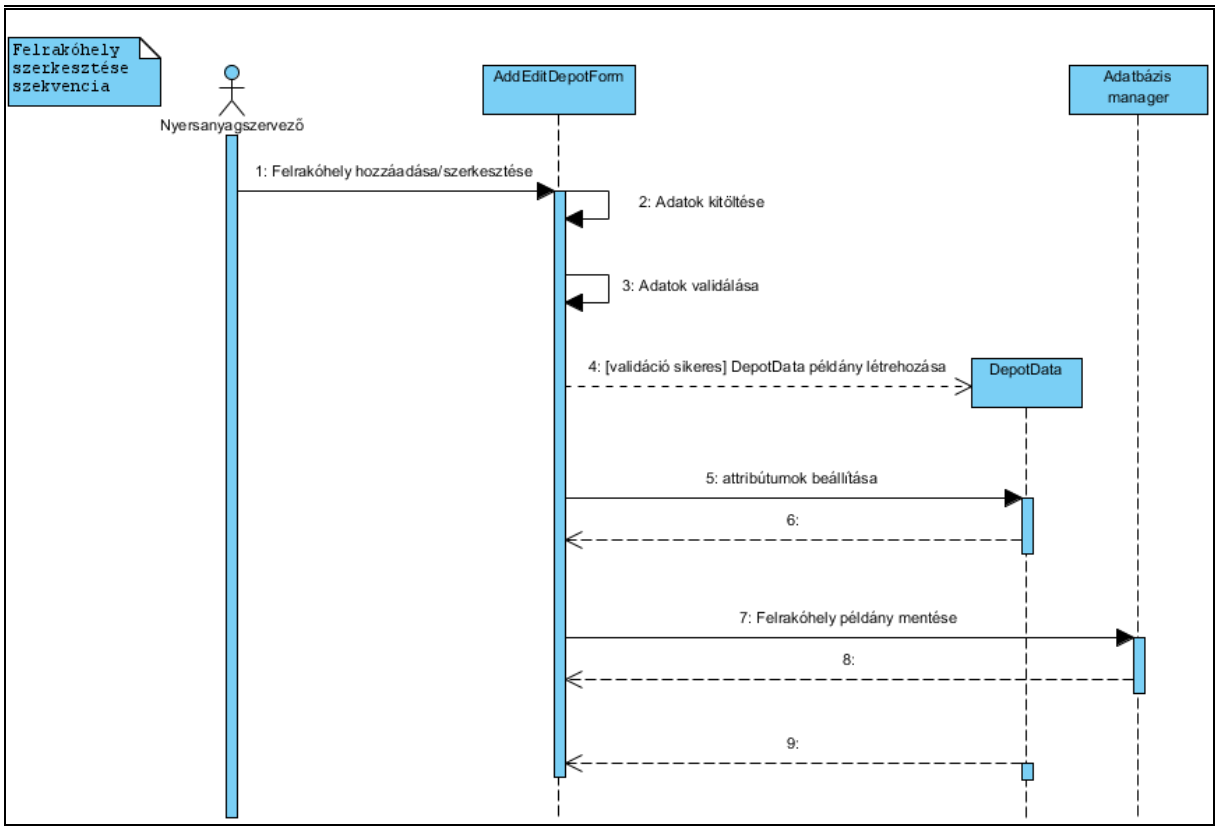
4.2.3.1 Osztálydiagramok

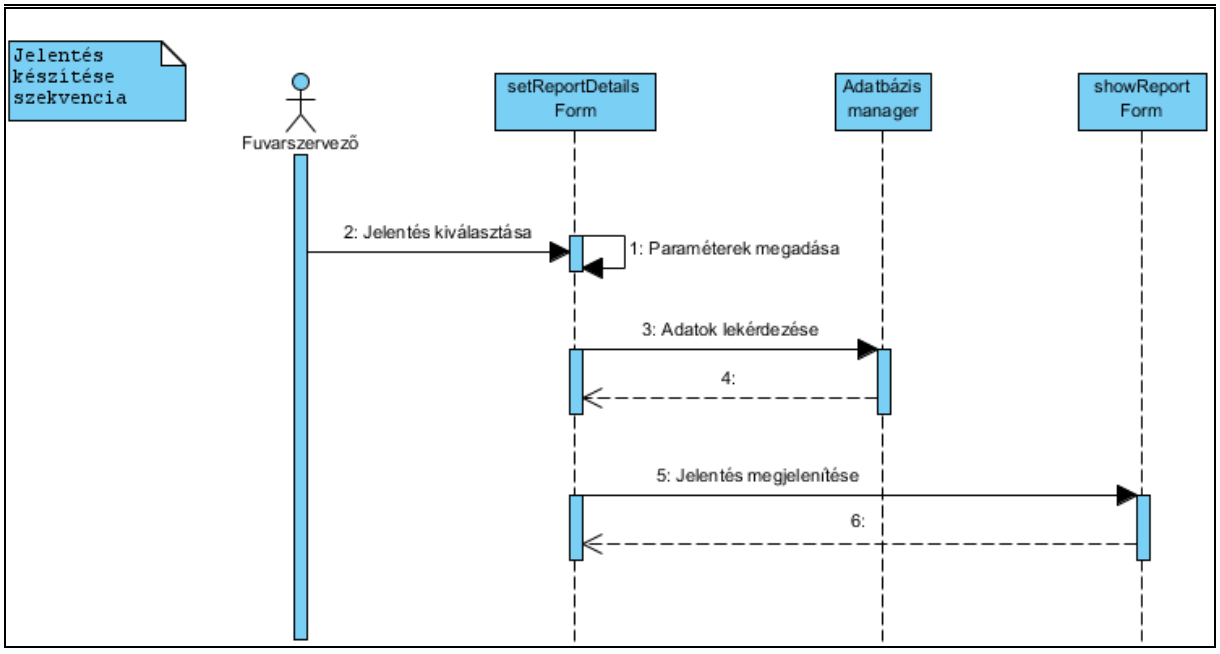
Az alkalmazás méreteiből adódóan több mint 40 osztályt tartalmaz. Ezen osztályok 95%-a különálló Form elemekként jelennek meg teljesen általános működéssel. Az osztályok közül különleges szerepe csupán a statikus ’Utils’ osztálynak van, ebben az osztályban gyűjtöttem össze az összes újrafelhasználható komponenst, melyre az alkalmazás bármely részén szükség lehet, és amely osztály a teljes alkalmazásból mindenhol elérhető.

Ebben az osztályban kaptak helyet az authentikációhoz, authorizációhoz szükséges függvények, a validálást végző függvények, a különböző konténer Control-okat feltöltő függvények, illetve az adatbázis ⟷ alkalmazás közötti konvertáló függvények.

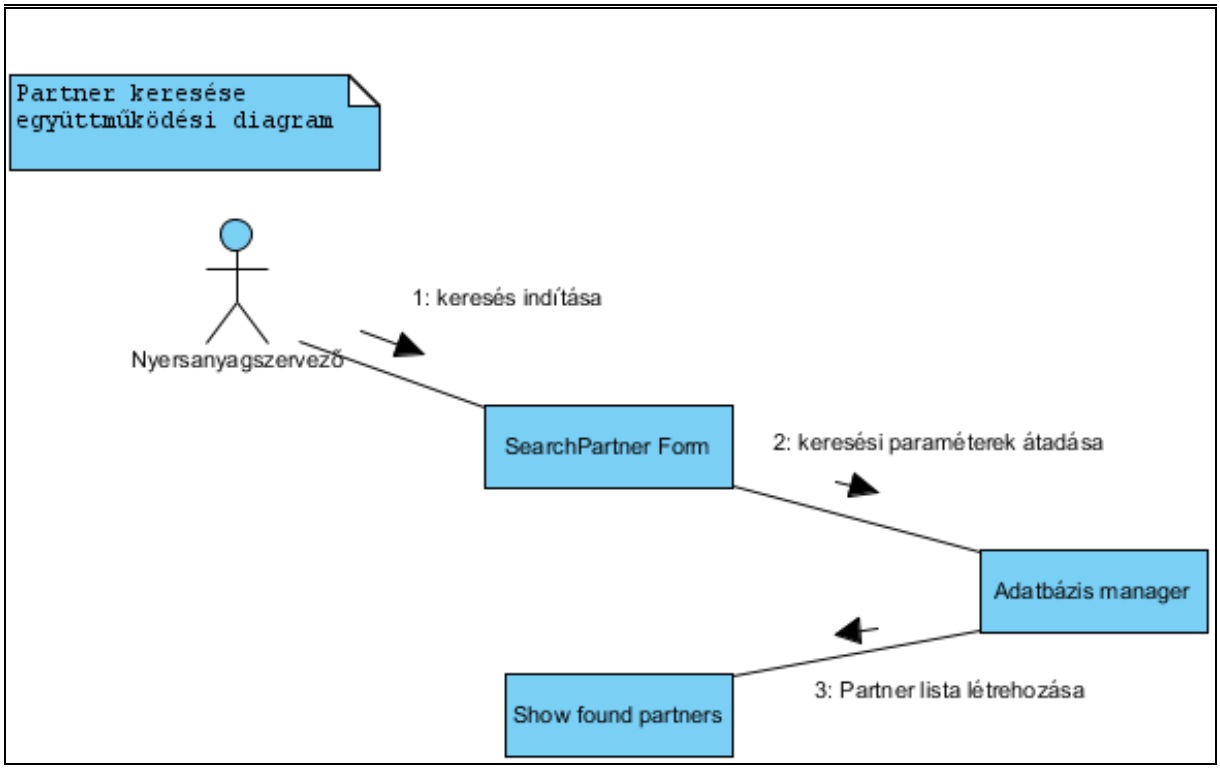


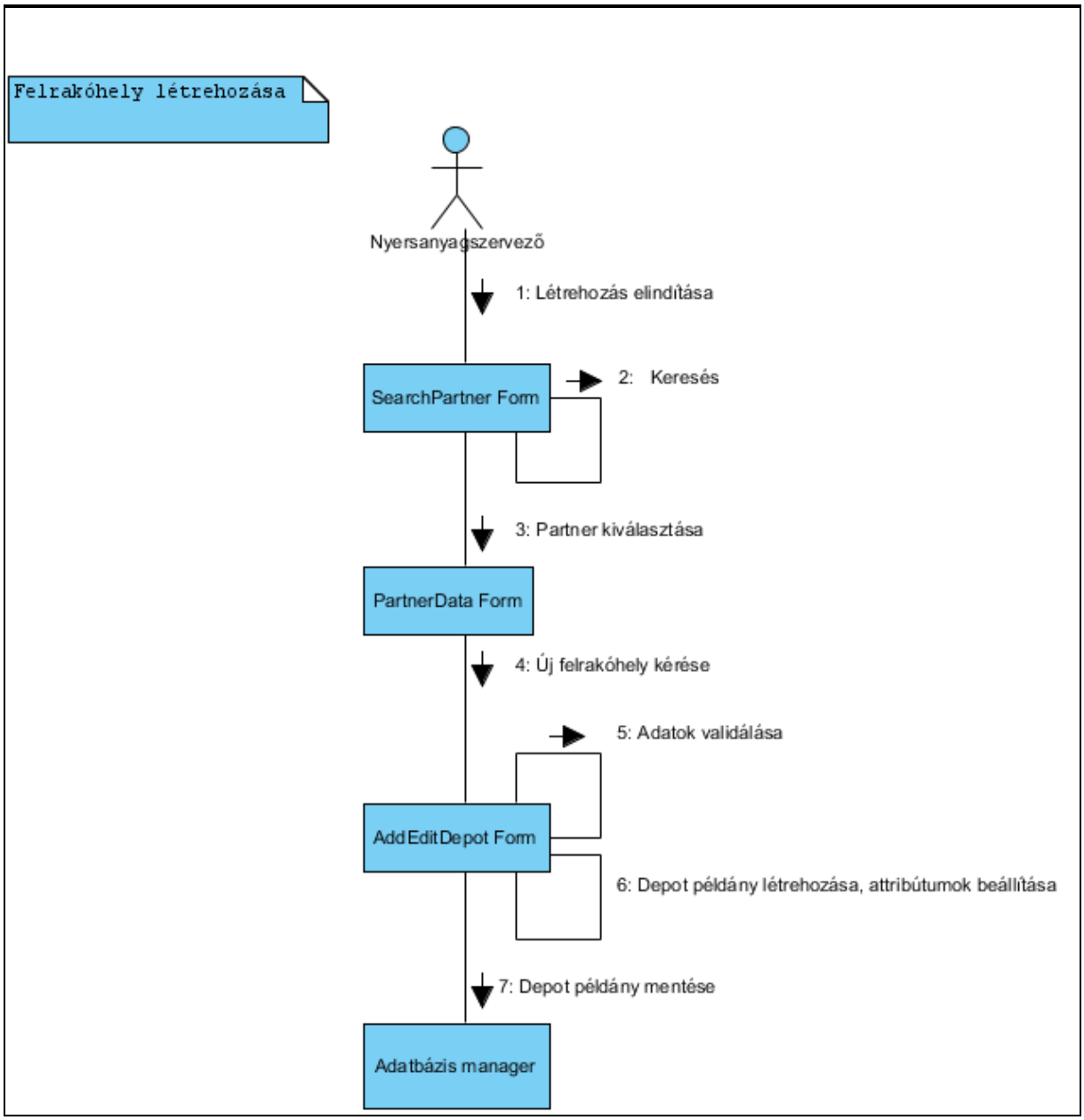
4.2.3.2 Szekvencia diagramok



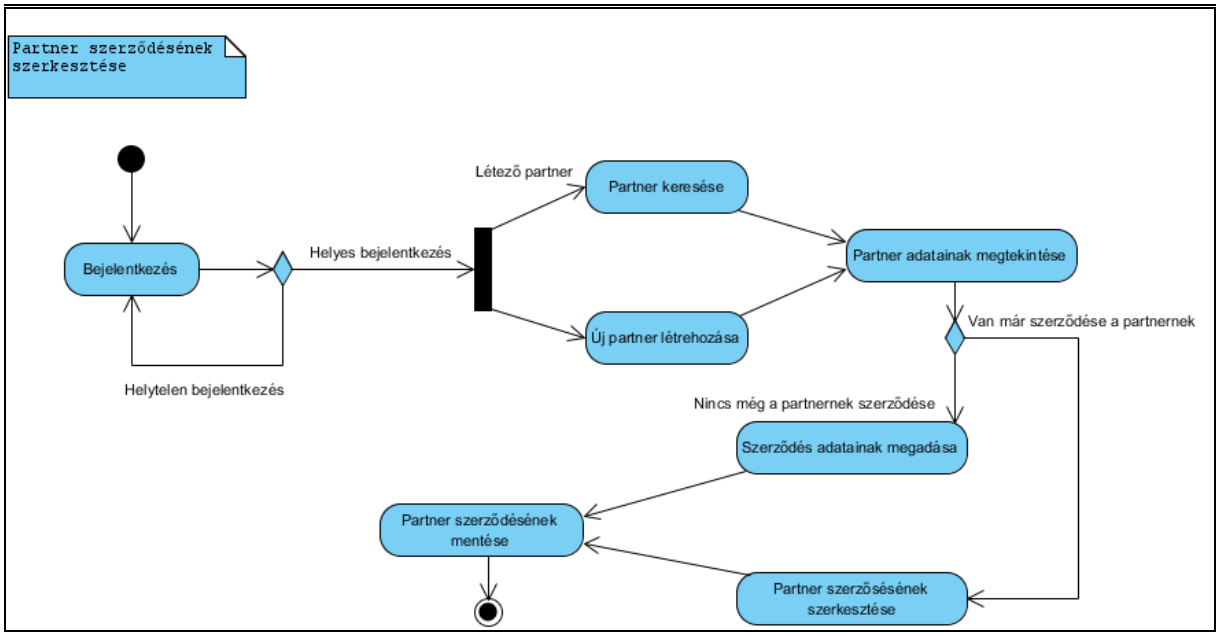


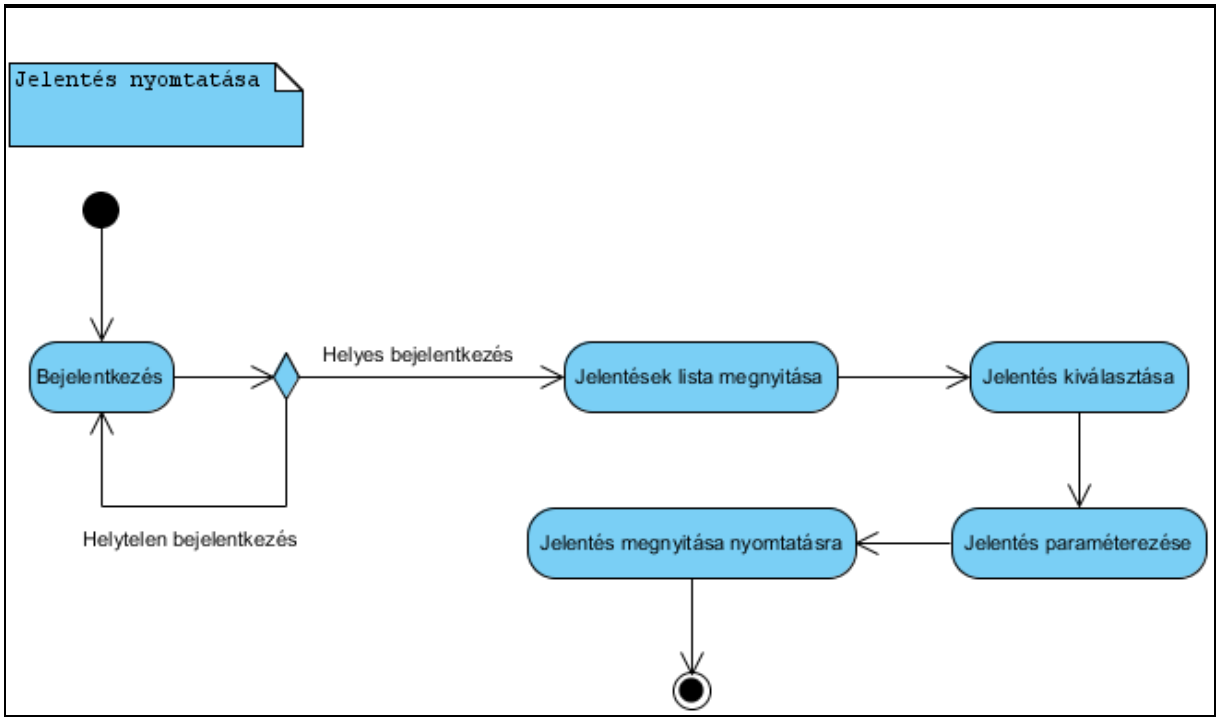
4.2.3.3 Együttműködési diagramok



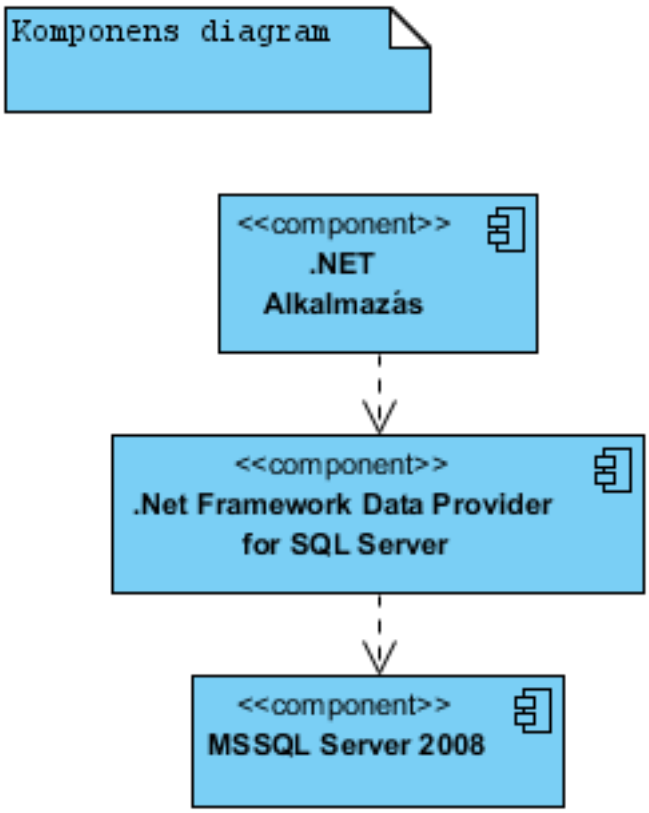

Felrakóhely létrehozása
Nyersanyagszervező
1: Létrehozás elindítása
SearchPartner Form
2: Keresés
3: Partner kiválasztása
PartnerData Form
4: Új felrakóhely kérése
5: Adatok validálása
AddEditDepot Form
6: Depot példány létrehozása, attribútumok beállítása
7: Depot példány mentése
Adatbázis manager

4.2.3.4 Aktivitás diagramok





4.2.3.5 Komponens diagram



4.3 Implementáció

4.3.1 Fejlesztői környezet

4.3.1.1 Microsoft SQL Server 2008

Azért esett a választásom erre az adatbázis kezelőre, mivel a Visual Studio 2008 integráltan támogatja, beépülő modulok nélkül is jól kezelhető. Az alkalmazás könnyű portabilitása érdekében az adatok nem egy adott SQL szerveren tárolódnak, hanem egy lokális .mdf kiterjesztésű fájlban az alkalmazás könyvtárrendszerében. Ugyanakkor a rendszer éles verziójában természetesen már erre a célra megfelelő adatbázis szerverre kerülnek az adatok. Ez az alkalmazás működésében nem okoz gondot, csupán a konfigurációs fájlban kell módosítani az adatbázis elérését.

4.3.1.2 Microsoft SQL Server Management Studio

Ez az alkalmazás egyszerűen kezelhető felületet nyújt az adatbázis manipulálásához, vizuális felületet ad a nézetek létrehozásához, ugyan a kapcsolatok beállítására vigyázni kell, ugyanis az eszköz előszeretettel alkalmaz CROSS JOIN-t illetve INNER JOIN-t.

A megkönnyített adatbázis kezelés mellett lehetőséget biztosít az adatbázisról biztonsági másolat készítésére, illetve sokrétű eszközt nyújt a különböző scriptkészítésekhez.

4.3.1.3 Visual Studio 2008 Professional

A Visual Studio a Microsoft több programozási nyelvet tartalmazó programozási termékcsomagja, amely az évek során egyre több új programnyelvvel bővült. A piacon ez az egyik legjobb (és legdrágább) fejlesztői csomag Windows-ra.

2007 végén jelent meg Visual Studio legutolsó, 2008-as verzióra (a kódneve Orcas). A kiadás egyik nagy újdonsága a multi-targeting, azaz szakítva az eddigi hagyományokkal - egy adott Visual Studio verzióval egy adott Framework verzióra lehetett szoftvert fejleszteni - a 2008 a .NET Framework 2.0-s, 3.0-s, illetve a fejlesztőkörnyezettel egy időben bétából véglegessé váló 3.5-ös verziójára is készíthetünk programokat.

A Visual Studio nagy előnye, hogy rengeteg előre kész modult tartalmaz, nem kell újra feltalálni azt, ami már működik. Jól kidolgozott API háttérrel rendelkezik, mely rengeteg példa forráskódot tartalmaz, ezzel is elősegítve az egyes függvények, osztályok működésének, használatának megértését. A felhasználói felületet pár mozdulattal létrehozhatjuk, karbantarthatjuk.

4.3.1.4 Bazaar verziókövető rendszer

A Bazaar egy elosztott verziókövető rendszer. Egyik legnagyobb előnye abban rejlik, hogy nem szükséges szerver a használatához, ugyanis támogatja a helyi repository létrehozásának lehetőségét is. Ez rendkívül hasznos egy olyan környezetben, ahol nincs mindig internetelérése az embernek, illetve egyszemélyes projectek esetén, mint az én esetemben is, sokkal kényelmesebb volt egy helyi repository-t létrehozni.

Egyszemélyes project esetén is nagyon hasznos egy ilyen eszköz alkalmazása, mivel pontosan visszakövethető a teljes kód minden változása, ami egy esetleges új funkcionalitás okozta problémánál könnyen áttekinthetővé teszi a két verzió közötti módosításokat, ezzel is megkönnyítve a hibakeresést.

4.3.2 Adatbázis implementációja

Az adatbázist az alábbi script-ek futtatásával lehet létrehozni:

- create_structure.sql - létrehozza az adatbázis struktúrát
- fill_data.sql - feltölti az adatbázist adatokkal

A fenti script-ek Microsoft SQL 2008 Server nyelven lettek létrehozva. A forrásfájlok a CD mellékleten megtalálhatóak. Az alábbiakban egy részlet látható a struktúrát létrehozó SQL állományból.

```
CREATE TABLE PartnerData (
  PartnerId DECIMAL  NOT NULL   IDENTITY ,
  Contractid DECIMAL    ,
  ContractTypesId DECIMAL    ,
  RegionsId DECIMAL    ,
  Name VARCHAR(255)    ,
  Place VARCHAR(100)    ,
  ZipCode VARCHAR(50)    ,
  Address VARCHAR(255)    ,
  PartnerNumber VARCHAR(50)    ,
  OrderNumber VARCHAR(50)    ,
  Administrator VARCHAR(100)    ,
  LetterAddress VARCHAR(255)    ,
  Owner VARCHAR(100)    ,
  ContactPerson VARCHAR(255)    ,
  ValidStart DATE    ,
  ValidEnd DATE    ,
  StatusFlag CHAR    ,
  PartnerDataComment VARCHAR(255)      ,
PRIMARY KEY(PartnerId)                ,
  FOREIGN KEY(RegionsId)
    REFERENCES Regions(RegionsId),
  FOREIGN KEY(ContractTypesId)
    REFERENCES ContractTypes(ContractTypesId),
  FOREIGN KEY(Contractid)
    REFERENCES Contract(Contractid));
GO

CREATE INDEX PartnerData_unique ON PartnerData (PartnerNumber, ValidStart, ValidEnd, StatusFlag);
GO
CREATE INDEX PartnerData_FKIndex1 ON PartnerData (RegionsId);
GO
CREATE INDEX PartnerData_FKIndex2 ON PartnerData (ContractTypesId);
GO
CREATE INDEX PartnerData_FKIndex3 ON PartnerData (Contractid);
GO
```

```
CREATE TABLE Contract (
  Contractid DECIMAL  NOT NULL   IDENTITY ,
  BankAccount VARCHAR(100)    ,
  KSHNumber VARCHAR(100)    ,
  KUJNumber VARCHAR(100)    ,
  TradeRegister VARCHAR(100)    ,
  TaxNumber VARCHAR(100)    ,
  TaxIdentifier VARCHAR(100)    ,
  MotherName VARCHAR(100)    ,
  IdentityCardNumber VARCHAR(100)    ,
  TownShipIdentifier VARCHAR(100)    ,
  MVHNumber VARCHAR(100)    ,
  Address VARCHAR(255)    ,
  ValidStart DATE    ,
  ValidEnd DATE    ,
  StatusFlag CHAR      ,
PRIMARY KEY(Contractid)  );
GO


CREATE INDEX Contract_unique ON Contract (ValidStart, ValidEnd, StatusFlag);
GO

CREATE TABLE DepotData (
  DepotId DECIMAL  NOT NULL   IDENTITY ,
  PartnerId DECIMAL  NOT NULL  ,
  RegionsId DECIMAL    ,
  DepotCode VARCHAR(50)    ,
  DepotName VARCHAR(255)    ,
  DepotPlace VARCHAR(100)    ,
  DepotAddress VARCHAR(255)    ,
  SiteDirectorName VARCHAR(100)    ,
  SiteDircetorPhone VARCHAR(45)    ,
  Weighing VARCHAR(45)    ,
  WeighingPlace VARCHAR(255)    ,
  isTransportByNotification CHAR    ,
  Comments VARCHAR(255)    ,
  ValidStart DATE    ,
  ValidEnd DATE    ,
  StatusFlag CHAR      ,
PRIMARY KEY(DepotId)            ,
  FOREIGN KEY(PartnerId)
    REFERENCES PartnerData(PartnerId)
      ON UPDATE CASCADE,
  FOREIGN KEY(RegionsId)
    REFERENCES Regions(RegionsId));
GO

CREATE INDEX DepotData_FKIndex1 ON DepotData (PartnerId);
GO
CREATE INDEX DepotData_unique ON DepotData (DepotCode, StatusFlag, ValidEnd, ValidStart);
GO
CREATE INDEX DepotData_FKIndex2 ON DepotData (RegionsId);
GO
```

4.3.3 Alkalmazás implementációja

Az alkalmazás C# nyelven készült a .Net FrameWork 3.5-ös keretrendszerében a Visual Studio 2008-as integrált fejlesztői környezetben. Az implementáció folyamán az alábbi szempontokat tartottam szem előtt:

Újrafelhasználhatóság

Fejlesztés során létrehoztam több ’User Control’ osztályt, melyek a többször ismétlődő GUI elemek csoportját és a hozzá tartozó forráskód másolgatást hivatott megelőzni. Ilyen ’User Control’ osztály például

- ’SaveLayout.cs’, mely egyforma megjelenést és eseménykezelést biztosít az adatbázis módosítást végző Form-ok számára.
- ’RadioFieldAllVsNothing.cs’, mely control egy két darab rádiógombból álló layout, amely az attribútumként megadott másik Control-t tiltja le vagy engedélyezi a gombok állása alapján. Erre a User Controlra a jelentéseknél van szükség, mikor a szűrési feltételeket beállítjuk. Így nem kell ugyanazt a kódot megírni többször, több feltétel esetén.

Általánosítás:

Ez a pont szorosan kapcsolatban áll az újrafelhasználhatósággal. Az alkalmazás részeit úgy terveztem meg, hogy néhány általános működés könnyen alkalmazható legyen rajtuk. Ilyen például a mentéskor frissülő kapcsolódó Form-ok, illetve a törlés előtti összes kapcsolódó Form bezárása.

Hogy ezt a működést megvalósítsam, készítettem egy ősosztály az összes Form számára, amelyben azonosító attribútumokat definiáltam, mint például partnerId, depotId. Ezen attribútumok minden megfelelő Form (PartnerForm, DepotForm) esetén kitöltésre kerülnek, így egy partner törlésekor az alkalmazás az összes nyitott ablakon végigiterál és el tudja dönteni, hogy az adott ablak a törlendő partnerhez tartozik-e és ha igen, akkor bezárja. Így elkerülhető az a probléma, hogy ugyan a partner törölve lett, mégis maradt olyan ablak nyitva, ahol felrakóhelye van, amit szerkesztve hibába futunk. Mentés utáni frissítés során pedig az összes ablak frissíti a tartalmát az adatbázisból (beleértve a keresőablakok keresési eredményét is), hogy aktuális értékek szerepeljenek.

Modulokra bontás:

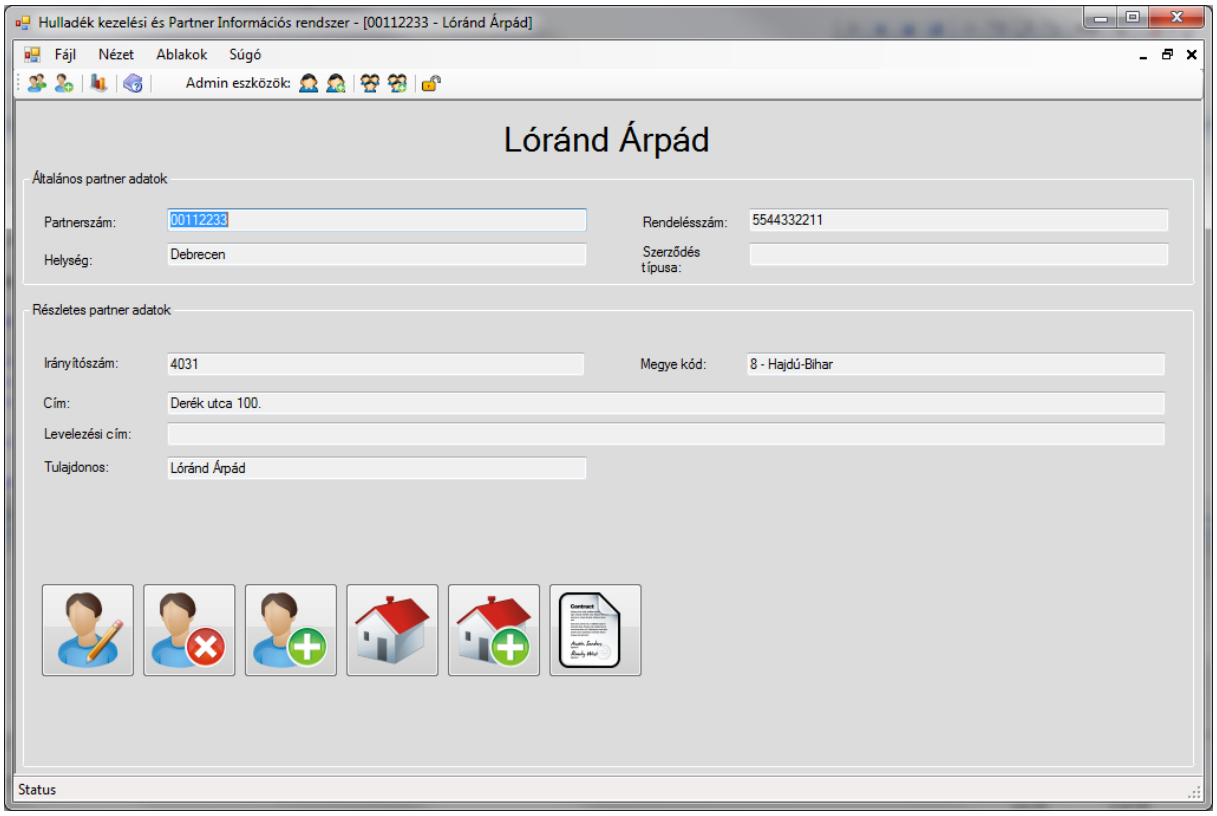
Az alkalmazás méreteit tekintve, illetve a jogosultságkezelés miatt is szükségszerű a modulok egyértelmű elkülönítése és azonosítása. Minden tevékenység külön modulba került, egyedül a szerkesztés és új létrehozása tevékenységek kaptak helyet ugyanabban a modulban

egyrészt praktikussági okokból, másrészt a jogosultságkezelés nem követelte meg az ennél mélyebb szintű elkülönítést.

Felhasználói felületek ergonómiája:

Nagyon fontos szempont az alkalmazásfejlesztés folyamatában, hogy jól áttekinthető, egységes, logikus felépítésű felületeket készítsünk, melyek használata közel egyértelmű, könnyen tanulható.

A felhasználóknak külön kérése volt felém, hogy minél több adatot láthassanak anélkül, hogy el kellene navigáljanak újabb ablakokra a részletek felfedése okán. Emiatt néhány Form kicsit robosztusra sikerült, sok adattal rajta. Ettől függetlenül törekedtem a jól strukturáltságra. Az alkalmazásban ’beszédes’ gombokat használtam, ezzel is növelve a felhasználói élményt.



Az alkalmazásban szereplő Form-ok három állományból állnak:

1. Form.cs – A Form-hoz tartozó osztályt, függvényeket tartalmazza
2. Form.Designer.cs – A Form GUI elemeit, azok elhelyezkedését és attribútumait tartalmazza

3. Form.resx – A .resx kiterjesztésû állományok tárolják az alkalmazások ûrlapjainak tartalmára vonatkozó információkat.

A jelentések felépítését leíró fájl egy XML fájl, ne tévesszen meg minket a kiterjesztése, mivel az .rdlc. Így report állományunk rendelkezik az XML azon jó tulajdonságával, hogy azt szabadon bármilyen szövegszerkesztővel manipulálhatjuk, ha szükség van rá. Nem kell ehhez fejlesztői környezet.

4.4 Verifikáció és validáció

4.4.1 Tesztelési környezet

Tesztelési környezetként Windows 7 operációs rendszer 64 bites változatát, .Net 3.5 SP1 framework-ot illetve Microsoft SQL Server 2008 Express Edition With Tools adatbázis kezelő rendszert használtam. Teszteléshez 4 felhasználót hoztam létre:

- Adminisztrátor - teljes jogosultsággal rendelkezik az alkalmazás felett
- Nyersanyagszervező - az adminisztrációs felületek kivételével teljes jogosultság
- Fuvarszervező - csupán megtekintési jogosultság, jelentéskészítéssel együtt
- Logisztika - fuvarszervezővel egyező jogosultságok

A továbbiakban az alábbi rövidítéseket fogom alkalmazni:

- Admin - Adminisztrátor
- NySzerv - Nyersanyagszervező
- FSzerv - Fuvarszervező
- Log - Logisztika

4.4.2 Funkcionális teszt

Az előbb létrehozott felhasználókkal végrehajtottam a funkcionális tesztelést, melynek során futás közben figyeltem a program reakcióit, hogy azok megfelelnek-e az elvárt működésnek. Az alábbiakban néhány tesztesetet, illetve a reakciókat foglaltam össze. A rendszer méretei miatt mindezt a teljesség igénye nélkül fogom most leírni.

Felhasználó	Végrehajtani kívánt tevékenység	Várt reakció	Kapott reakció
admin	Új felhasználó felvitele ’teszt’ névvel a fuvarszervezői csoportba	Az új felhasználó sikeresen létrejön	OK
admin	Logisztika csoporthoz partner módosítása jog hozzáadása	A jogosultság sikeresen hozzáadódik a Logisztika csoporthoz	OK
admin	Új felhasználó felvitele ’teszt’ névvel az Adminisztrátori csoportba	A felhasználó nem jön létre, mivel már létezik ilyen felhasználó	OK
admin	’teszt’ felhasználó inaktívvá tétele	A ’teszt’ felhasználó inaktívvá válik	OK
teszt	Bejelentkezés	A ’teszt’ felhasználó nem tud bejelentkezni, mivel nem aktív	OK
admin	’Lóránd Árpád’ partner megtekintése	A kért partner adatai megjelennek	OK
NySzerv	’Jóska Béla’ partner létrehozása	Az új partner létrejön, az adataival megnyílik a partner ablak	OK
NySzerv	’Jóska Béla’ partnerhez új felrakóhely felvitele ’Jóska juhtelepe’ néven	Az új felrakóhely létrejön, az adataival megnyílik a felrakóhely ablak	OK
NySzerv	’Jóska Béla’ partnerhez szerződés adatok megadása	A szerződés adatok hozzáadásra kerülnek, megnyílik a szerződés megtekintése ablak az új adatokkal	OK
NySzerv	Jelentés készítése a járatokról	A megadott szűrési feltételeknek megfelelően megtörténik az adatok legyűjtése és megjelenik a jelentés, ami ezután nyomtatható, exportálható	OK
NySzerv	’Jóska juhtelepe’ felrakóhelyhez új járatkód (’C’) rendelése	A szerkesztésre megnyitott felrakóhely, majd bővített járatkód lista mentésre kerül. A járatkódok között megjelenik az új ’C’ járatkód is	OK
NySzerv	Új partner ’Kiss Elek’ létrehozása	Az új partner sikeresen létrejön	OK
NySzerv	Új felhasználói csoport létrehozása	Sikertelen, a felhasználó nem is éri el az adott ablakokat, mivel nincs jogosultsága hozzá	OK
FSzerv	’Jóska Béla’ partner felrakóhelyeinek böngészése	A felhasználó sikeresen tudja böngészni mind a partnert, mind a felrakóhelyeit	OK
FSzerv	’Jóska Béla’ partnerhez új felrakóhely hozzáadása	Sikertelen, a felhasználó nem is éri el az adott felületet, mivel nincs jogosultsága hozzá	OK

Fszerv	’Jóska Béla’ partner címének módosítása	Sikertelen, a felhasználó nem is éri el az adott felületet, mivel nincs jogosultsága hozzá	OK
FSzerv	Jelentés készítése az állattartó telepekről	A megadott szűrési feltételeknek megfelelően megtörténik az adatok legyűjtése és megjelenik a jelentés, ami ezután nyomtatható, exportálható	OK
Log	Új partner hozzáadása ’Márton Áron’ névvel	Sikertelen, a felhasználó nem is éri el az adott felületet, mivel nincs jogosultsága hozzá	OK
Log	’Kiss Elek’ partner adatainak megtekintése	A partner adatai megtekinthetők	OK
Log	’Kiss Elek’ partner törlése	Sikertelen, a felhasználó nem is éri el az adott felületet, mivel nincs jogosultsága hozzá	OK
Log	Jelentés készítése a járatokról	A megadott szűrési feltételeknek megfelelően megtörténik az adatok legyűjtése és megjelenik a jelentés, ami ezután nyomtatható, exportálható	OK

A funkcionális tesztek sikeresen lezajlottak, a megadott tesztesetek minden esetben a várt eredménnyel zárultak. Az alkalmazás most már átadható a felhasználóknak tesztelésre.

5 Összefoglalás

A diplomamunkámban célul kitűzött rendszermodulok fejlesztése sikeresen lezárult. Végighaladtam a fejlesztés menetén, felmértem a követelményeket, elemeztem azokat és tárgyaltam a megvalósítási lehetőségekről a felhasználókkal. Ezalatt a folyamat alatt sokszor szembesültem azzal, hogy sok esetben a felhasználóknak sincs konkrét elképzelésük arról, hogy pontosan mit is szeretnének. Voltak ötletek, kívánságok, de ezek csak a segítségemmel együtt forrtak ki használható követelményekké. A követelmények rögzítése után elvégeztem a szükséges tervezési lépéseket. A tárolandó adatok alapján megterveztem az alkalmazás alapjául szolgáló adatbázis struktúrát, majd pedig az UML diagramok segítségével rögzítettem az alkalmazás terveit, melyek később alapjául szolgáltak az implementálás folyamán. Az implementálás során végig szem előtt tartottam a legfontosabb fejlesztési szempontokat és törekedtem arra, hogy bármilyen későbbi módosítás, kiegészítés egyszerűen végrehajtható legyen. Az elkészült alkalmazást kellő alapossággal teszteltem, a fejlesztés közben fellépő hibákat kijavítottam. A végső teszten az alkalmazás 100%-os eredményt ért el.

Mindezek alapján úgy ítélem meg, hogy a kitűzött célokat elértem. Kifejlesztettem egy jól strukturált, alkalmazást, melynek használatával nagy terhet veszek le a Nyersanyagszervezői és Fuvarszervezői munkakörben dolgozók válláról.

A jövőben a rendszer további fejlesztésén fogok dolgozni, illetve a hiányzó modulok implementálásán, de ez már meghaladja a diplomamunkám kereteit.

6 Irodalomjegyzék:

- Dr. Juhász István: Rendszerfejlesztés technológiája előadás jegyzet
- Maksimchuk, Robert A. –Naiburg, Eric J. (2006): UML földi halandóknak. Kiskapu Kiadó, Budapest
- Matthew A. Stoecker and Steven J. Stein withTony Northrup: Microsoft .Net Framework 2.0 Windows-Based Client Development
- http://www.atev.hu
- http://weblogs.asp.net/scottgu/archive/2007/05/19/using-linq-to-sql-part-1.aspx
- http://hu.wikipedia.org/wiki/Microsoft_Visual_Studio
- http://bytes.com/topic/c-sharp/answers/653729-combobox-usage
- http://www.csharp.hu/default.aspx?page=16&articleid=61
- http://en.wikipedia.org/wiki/Microsoft_SQL_Server
- http://www.microsoft.com/hun/technet/tc/?id=6d7b8cb7-e513-4394-b1c9-2af934d830b3
- http://en.wikipedia.org/wiki/Bazaar_(software)
- http://hu.wikipedia.org/wiki/Unified_Modeling_Language

7 Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani:

- Vágner Anikónak, amiért elvállalta témavezetésem és tanácsaival, szaktudásával segítette diplomamunkám végső formáját, tartalmát megalkotni.
- Családomnak, akik kitartóan támogattak tanulmányaim során.
- Végül pedig köszönöm páromnak, Barabás Kingának a lelki támogatást, a folyamatos unszolást és a rengeteg türelmet, amit felém tanúsított ezalatt az idő alatt.