SOPRONI EGYETEM

SIMONYI KÁROLY MŰSZAKI, FAANYAGTUDOMÁNYI ÉS MŰVÉSZETI KAR

INFORMATIKAI ÉS GAZDASÁGI INTÉZET

Raktározási és cikkcsoportok közötti kapcsolat kialakítása Microsoft Dynamics NAV környezetben

Tukovits Balázs

Szakdolgozat

Konzulens: Dr. Szabó László

2020. december 4.

Soproni Egyetem, Simonyi Károly Műszaki, Faanyagtudományi és Művészeti Kar

Informatikai és Gazdasági Intézet

9400 Sopron, Bajcsy-Zs. u. 4.

SZAKDOLGOZAT FELADAT

Szakdolgozatot készítő neve:	**Tukovits Balázs** gazdaságinformatikus BSc hallgató
A szakdolgozatot készítő Neptun kódja:	RFGWRL
Szakdolgozat címe:	**Raktározási és cikkcsoportok közötti kapcsolat kialakítása Microsoft Dynamics NAV környezetben**
Intézeti konzulens:	**Dr. Szabó László,** egyetemi docens
Külső konzulens(ek):	**Koncz Vera,** Team Manager, Cosmo Consult Kft
A dolgozat kódja	SKK-INGA-9-2020-SZ

Kis- és középvállalatok számára piacvezető ERP rendszer a Microsoft Dynamics NAV. Népszerűségében fontos szerepet játszik a flexibilitása, ha a felhasználónak szüksége van a beépítetteken kívül további funkciókra, szakemberek segítségével egy speciális nyelven, saját fejlesztőkörnyezetben megvalósíthatja azokat. Szakdolgozatomban egy olyan kiegészítőt implementálok, amely segíti és gyorsítja a raktárosok munkáját a cikk betárolásától annak kitárolásig.

Elvégzendő feladatok

1. Navision adatbázis módosítása
2. Gyári táblák, Page-ek módosítása
3. Új Page-ek létrehozása
4. Eseménykezelés megvalósítása
5. Az adatbázis teszt adatokkal való feltöltése
6. Tesztelés

Beadási határidő: 2020. december 04. 12:00

Sopron, 2020.09.25.

Prof. Dr. Magoss Endre
dékán

Dr. Bednárik Éva
intézetigazgató

P.H.

NYILATKOZAT

Alulírott **Tukovits Balázs** (neptun kód: **RFGWRL)** jelen nyilatkozat aláírásával kijelentem, hogy **Raktározási és cikkcsoportok közötti kapcsolat kialakítása Microsoft Dynamics NAV környezetben** című

szakdolgozat

(a továbbiakban: dolgozat) **önálló munkám**, a dolgozat készítése során betartottam a szerzői jogról szóló 1999. évi LXXVI. tv. szabályait, különösen a hivatkozások és idézések tekintetében.

Hivatkozások és idézések szabályai:
Az 1999. évi LXXVI. tv. a szerzői jogról 34. § (1) és 36. § (1) első két mondata.)

Kijelentem továbbá, hogy a dolgozat készítése során az önálló munka kitétel tekintetében a konzulenst, illetve a feladatot kiadó oktatót **nem tévesztettem meg.**

Jelen nyilatkozat aláírásával tudomásul veszem, hogy amennyiben bizonyítható, hogy a dolgozatot nem magam készítettem, vagy a dolgozattal kapcsolatban szerzői jogsértés ténye merül fel, a Soproni Egyetem megtagadja a dolgozat befogadását és ellenem fegyelmi eljárást indíthat.

A dolgozat befogadásának megtagadása és a fegyelmi eljárás indítása nem érinti a szerzői jogsértés miatti egyéb (polgári jogi, szabálysértési jogi, büntetőjogi) jogkövetkezményeket.

Sopron, 2020.december 04.

……..……..

hallgató

Tartalom

Absztrakt

Szakdolgozatom témáját munkaadómtól kaptam. A Microsoft Dynamics NAV egy integrált vállalatirányi rendszer (ERP-rendszer), kis- és középvállalatok számára. A cég end-to-end megoldásokkal szolgál ügyfeleinek, kényelmesebbé, könnyebbé téve azok munkáját. A feladatom is egy ilyen új, plusz segítő funkció elkészítése volt. A NAV raktárait (Locations) és cikkcsoportjait (Item Category) alapértelmezetten nem lehetséges felparaméterezni, csak a raktár zónáit. Ezért kellett készítenem egy olyan megoldást, amiben lehetőség van raktárakat és a cikkcsoportokat paraméterekkel ellátni. Ezen jellemzők alapján kell tudnom összekapcsolni a raktárakat és a cikkcsoportokat egy algoritmus segítségével. Ez a raktározási problémák minimalizálása miatt nagyon hasznos megoldás. Az összekapcsolások könnyen módosíthatók, mind felvételkor, mind pedig törléskor. Ezt jól beállított jogosultsági szintekkel teszem letisztultabbá.

Abstract

I recieved the topic of my thesys from my employer. Microsoft Dynamics NAV is an integrated ERP (Enterprise Resource Plannaing) system for small- and minddle-sized companies. The company gives end-to-end solutions for it's customers, to make their work more comfortable and easier. My task was to make a new, helping feature. It's not possible to attach parameters to the standard NAV Locations and Item Categories, just the Location Zones. That's why i had to make a sulotion, in which we can assign parameters to our Locations and Item Categories. Based on these parameters, we are able to connect the Locations and Item Categories via an algorithm. This solutions is really useful in minimizing the storing problems. The connections can be easily modified, as well as when we assign, or when we delete connections. I make it transparent with proper access management.

1. Bevezetés

A fejezetben bemutatom a szakdolgozatomhoz köthető motivációkat, a gyakornoki munkámat a Cosmo Consult Kft-nél (továbbiakban Cosmo) és az elmúlt félévek munkáját. Témalabor tárgy kereteim belül ismerkedtem meg jelenlegi munkaadómmal, egy általuk meghirdetett témát választottam. Ezt bővítettem később Önálló Laboratórium tárgyam keretein belül, ahol közel kerültem a végleges megoldáshoz. A raktárak és cikkcsoportok közötti kapcsolatot a NAV és a Business Central (továbbiakban: BC) sem kezeli, ugyanis csak a raktárak zónáit van lehetőség paraméterekkel ellátni. Sem magát a teljes raktárat, sem a cikkcsoportot nem tudjuk felparaméterezni. A kapcsolat kialakítására azért van szükség, hogy gyorsíthassuk a raktári folyamatokat (ki és betárolás, kiszedés, komissiózás stb). Ezzel az új kiegészítővel nagy mértékben növelhető a raktár hatásfoka, könnyebbé tehető a raktári alkalmazottak munkája. Gördülékenyebb lehet az ügyintézés, a hibás letárolásokból történő fennakadások száma is csökkenhet. A korábbi Témalabor tárgy bemutatója után kaptam ajánlatot a Cosmotól, lehetőség nyílt gyakornokként a cégnél elhelyezkedni. Témalabor és Önálló Laboratórium (továbbiakban: Önlab) tárgyaim szolgáltak a témaválasztás alapjául. Először órai, majd az előbb említett tárgyak keretében kezdtem el megismerkedni a Cosmo által forgalmazott vállalatirányítási rendszerrel (továbbiakban: ERP), így könnyebb volt elkezdeni gyakornoki feladataimat.
Először a motivációimat szeretném bemutatni.

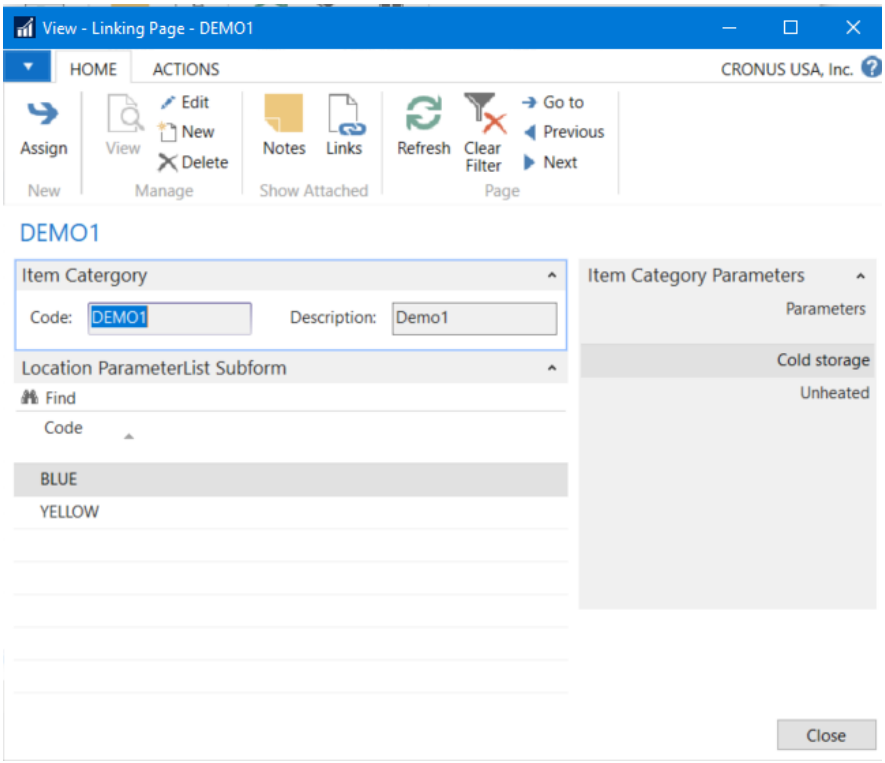
1.1. Motivációk

Középiskolai éveimben nem tudtam, mivel szeretnék foglalkozni. Ismerőseim ajánlására jelentkeztem a gazdaságinformatikus képzésre. Itt is nehéz helyzet elé kerültem. A programozás elég távol állt tőlem a kezdetekben, így a 3. féléves Vállalatirányítási rendszerek (továbbiakban: VIR) segített a döntésemben. Tetszett az informatika és a közgazdaságtan ilyen mélységű kapcsolata, így ekkor döntöttem el, hogy ebbe az irányba szeretnék specializálódni. Döntésemet az 5. félévesben meghirdetett szabadon választható tárgy pecsételte meg, ERP rendszerek ipari alkalmazása és fejlesztése néven indult, és a Cosmo tartotta. Itt lehetőségem volt megismerkedni a cég vállalati kultúrájával, szakértelmével és felfogásával. A tárgy keretein belül jobban beleláthattam a NAV működésébe és struktúrájába, ami óriási segítséget nyújtott Témalabor és Önlab dolgozat elkészítésében.
A NAV használatával a VIR tárgy gyakorlat keretében ismerkedtem meg. Itt felhasználó szintű ismereteket szereztem, majd a korábban említett ERP fejlesztés tárgy keretében fejlesztői oldalról is beleláthattam a rendszerbe. Ezeket felhasználva készítettem el Témalabor

dolgozatomat. Választott témámat hasznosnak látta a Cosmo, hiszen erre a megoldásra szükségük van az üzleti életben. Megtetszett a cégben uralkodó légkör és mentalitás, így biztossá vált számomra, hogy szeretnék a Cosmo család tagja lenni. Szeretném feladatomat minél teljesebben megvalósítani és az üzleti életben átültetni más verziókra is.
Témaválasztásomat egy további, belső motiváció is irányította. Korábban is érdeklődtem a raktározás és logisztika iránt, így a témaválasztás egyértelművé vált.

1.2. Megoldandó feladat ismertetése

Célszerűnek tűnt számomra több kisebb egységre bontani a megoldandó feladatomat, így könnyebb volt átlátni a megoldás menetét és sokkal könnyebb volt látnom, mit kell még létrehoznom a rendszerben.
Témalabor keretein belül megismerkedtem a programmal, a logisztikai folyamatokkal, a NAV raktározási folyamataival (raktárhelyek, zónák stb), elkészítettem egy kezdetleges adatbázis módosítás tervezetet és az összehasonlító algoritmusom pszeudó kódját. Önlab dolgozatomban már a megvalósításon dolgoztam. Először az adatbázis Táblákat (Table) hoztam létre, ezek után a Page-eket. Ezek alapjául az általam létrehozott táblák szolgáltak. Több Page-re volt szükségem, hiszen a NAV-ban gyakran használt Fej-soros (Page- Subpage) kapcsolatot kellett kialakítanom. Végül pedig az összehasonlító algoritmust írtam meg.
A Page futtatásával a Card (Karton típusú Page, továbbiakban: Karton) részen látnunk kell a kiválasztott cikkcsoportot, míg a sorokban a hozzá rendelt Raktárak kódjait. A Factboxban (oldalmenü) a cikkcsoportunkhoz tartozó paramétereket kell megjeleníteni. Ezek tervezését, létrehozását a későbbiekben pontosan fogom ismertetni.



1. ábra: Lehetséges megoldás kinézete

Legfontosabb feladataim is az előbb említettek voltak. A NAV-ban minden egymásra épül, integrált rendszerről lévén szó. Ha bármely pontban csak egy minimális hiba lép fel, már hibás az összekapcsolás, vagy rossz a megjelenítés. Fontos a jó tábla struktúra kialakítása, a helyes kulcsválasztás. Ezek alapján már a jól megírt algoritmus megtalálja az összetartozó párokat, amiket azután a felhasználó számára meg is tudunk majd jeleníteni.

A jogosultságkezelés mellett itt sem mehetünk el szó nélkül. A Cosmoval egyeztetve kerülnek majd beállításra a jogosultsági szintek. A felhasználó csak akkor tudja futtatni az általam készített oldalt, ha az adott Role kapott hozzá jogosultságot. A NAV struktúrájából adódóan a felhasználókat Role-ok szerint csoportosítja, és mindenki a saját Role-jának megfelelő RTC-t (Role Tailored Client) tudja csak futtatni.

Fontos kiemelni, ez a megoldás nem cégspecifikus. Az általam elkészített kiegészítőt bármely ügyfelünk rendszerébe tudjuk majd implementálni. A Cosmonál a „dobozos szoftvert” szabjuk az ügyfél igényeire, amire szüksége van, azt belefejlesztjük, így ez a funkció igény szerint kerül bele az ügyfél rendszerébe.

1.3. Felhasznált szoftver

Ebben a fejezetben bemutatom a felhasznált szoftvert, amivel az elmúlt félévekben és most munkám során foglalkozom. Az üzleti életben további Microsoft ERP rendszereket is használunk, a NAV-nak több verziója van, továbbá használunk még BC-t is, melynek szintén több verziója van, és nem utolsó sorban AX-et, amely a Microsoft nagyvállalati ERP megoldása.

A megoldásomhoz Dynamics NAV 2016-ot használtam, iskolai és oktatási célra kaptunk a rendszerhez licencet. Ezt egyeztettem a Cosmoval is, így használhattam ezt a verziót. Ez egy On Premsie verzió, tehát a kliens és a szerver is a saját gépemen fut. Így a kliens mellett több további kapcsolódó szoftver is települ a számítógépre.

Ezek a következők:

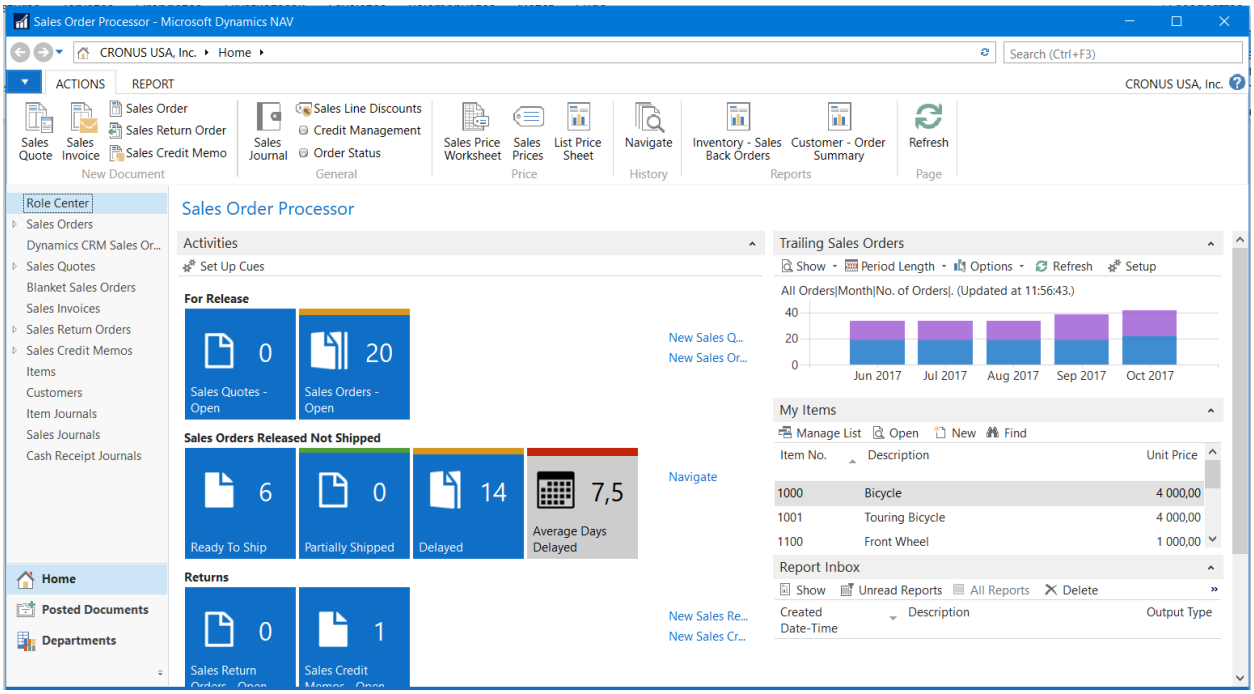
- Dynamics NAV 2016
- Dynamics NAV Development Environment
- Dynamics NAV Development Shell
- Dynamics NAV Administration
- Dynamics NAV Administration Shell

A NAV a Microsoft kis- és középvállaltok számára létrehozott ERP megoldás, ami az egyik legjobb a piacon. Évről évre, verzióról verzióra egyre jobban, biztonságosabban és stabilabban működik. Az elmúlt években jelent meg Cloud alapú testvére a BC, ami jelenleg az egyik legjobb termék a piacon.

Szakdolgozatom a fent említett programok közül a Klienst (NAV 2016) és a fejlesztői környezetet használtam fel a feladat teljesítésére. Nagy mennyiségű és kiváló minőségű dokumentáció áll rendelkezésre ehhez a szofvercsaládhoz, ami nagyban könnyíti/könnyítette a munkámat.

Microsoft Dynamics NAV 2016

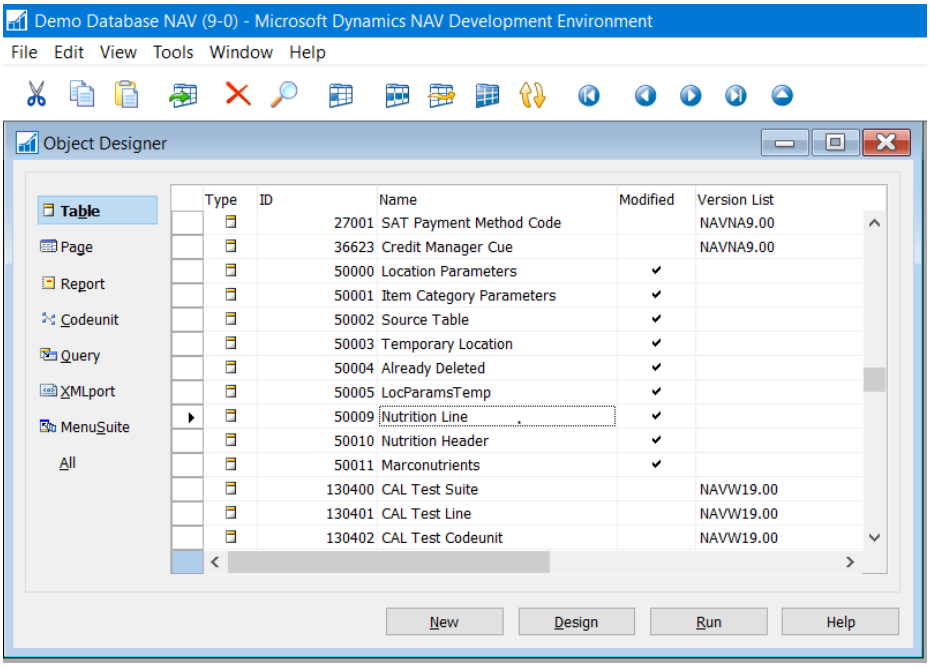
A NAV egy többnyelvű, több pénznemet is kezelni tudó üzleti megoldás, ami rengeteg vállalatot segít világszerte. Ez egy integrált ERP rendszer, segít a könyvelésben, pénzügyekben, ellátási láncban és a működésben is. Minden a vállalat életében nélkülözhetetlen dologra megoldást nyújt. Dinamikusan fejleszthető üzleti szoftver, ami képes a változó üzleti élethez igazodni. Előnye, hogy jól kommunikál és működik együtt a Microsoft Office 365 megoldásaival (például Reporting – Word, Excel, PowerBI stb). Megkönnyíti alkalmazottak mindennapi munkáját, akár az irodából, akár otthonról.



2. ábra: NAV 2016 Kliens

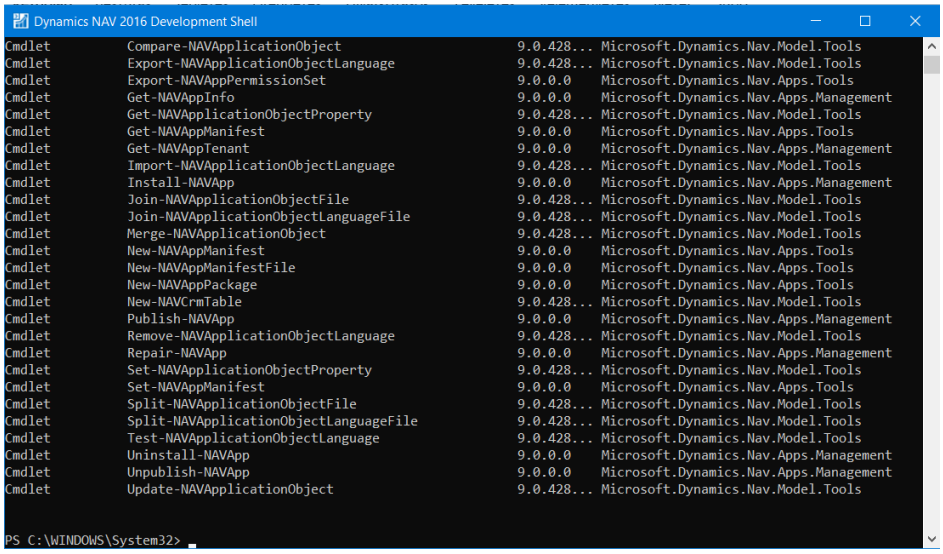
Dynamics NAV Development Environment

A Development Environment a NAV fejlesztői környezete. Ebben tudunk új megoldásokat létrehozni C/AL (Client/Application Language) nyelven. A C/AL egy Pascal alapú nyelv. Objektum alapú, de nem objektum orientált(OO). Az Object Designerből tudjuk kiválasztani az aktuálisan módosítani kívánt objektumot. Ezek lehetnek: Table, Page, Report, Codeunit (OO programozásban az osztálynak feleltethető meg), Query, XMLPort és MenuSuite.



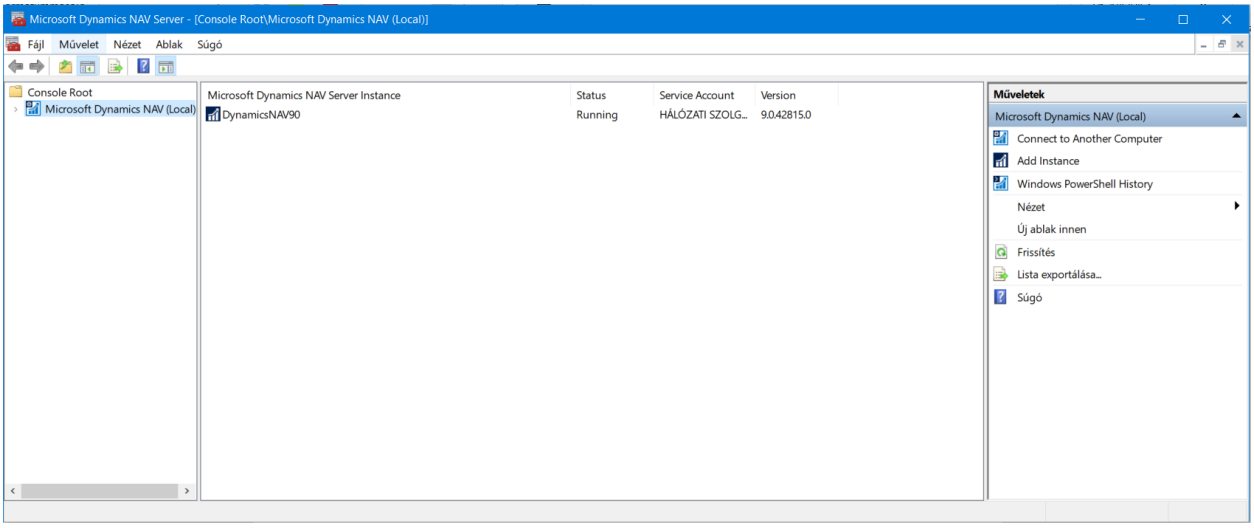
3. ábra: NAV Development Environment

A NAV telepítésekor további alkalmazások is települnek a számítógépünkre. Ezek a Dynamics NAV Development Shell és Dynamics NAV Administration. A Development Shell segítségével hozzáférhetünk PowerShell cmdlet-ekhez (parancsmagokhoz). A cm3det egy PowerShell-ben használatos egyszerű parancs. Egyszerű műveleteket hajtanak végre, általában egy Microsoft .NET alapú objektumot adnak vissza a parancs lefutása után, mellyel a következő parancs/hívás tud dolgozni.



4. ábra: Dynamics NAV Development Shell

A Dynamics NAV Administration Tool egy Microsoft Management Consol (MMC). Segítségével láthatunk bele és kezelhetjük a szerver instance-okat.



5. ábra: Microsoft Dynamics NAV Administration

1.4. Összegzés

Összefoglalásként egy rövid összegzéssel zárom a fejezetet. A fejezetben ismertettem motivációimat, a feladatválasztás hátterét. Az 1.2.-es fejezetben bemutattam a megoldandó problémát. Emellett a feature eddig elkészült fázisait, elemeit és ezek elkészítési lépéseit villantottam fel.

Bemutattam a NAV 2016 programcsomagba tartozó szoftverek és ezek kapcsolatát, melyek segítségével létre tudtam hozni a Cosmo által kért kiegészítőt. A megoldás maga nem triviális. Az elején több nehézségbe is ütköztem, de Horváth-Lukics Martin és Koncz Vera segítettek megérteni a NAV logikáját, így elkészülhetett a fejlesztés.

A soron következő egységben bemutatom a raktározás alapfogalmait és ezek implementációját a NAV rendszerben.

2. Raktározás

A logisztikai folyamatok között a raktározás egy olyan folyamat, amely a vállalaton belül kialakult ellátási lánc szerves részét képezi, az egyes részfolyamatok között helyezkedik el. Az éppen adott gyártási folyamat outputját kezeli, tárolja le a lehető legmegbízhatóbb és legbiztonságosabb módon.

Mivel minden vállalat célja a profitmaximalizálás, így a vállalat - mind házon belül, mind a vevői irányába - törekszik a JIT (Just In Time) elv teljesítésére, ami azt jelenti, hogy két folyamat között a várakozási idő nulla vagy elenyésző mértékű. Ennek a feltétele, hogy a rendszer képes legyen többletmennyiség felhalmozására. Itt kerülnek előtérbe a raktárak.

Mint mindenhol, ebben a témában is vannak megfogalmazott, általánosan elfogadott alapelvek, melyekre minden raktár törekszik:

- egységrakományokat igyekszik kialakítani, amik legjobban illenek a raktár és a szállítmányozók struktúrájába,
- jó terület gazdálkodás – minél jobban kihasználja a rendelkezésére álló teret,
- minél kevesebbet kelljen az árut mozgatnia,
- a mozgatott áru minél gördülékenyebben, minél hamarabb jusson el a helyére,
- garantálni tudja a benne a tárolt áruk biztonságát,
- ügyeljen környezetére.

És a legfontosabb, hogy mindezt a lehető legalacsonyabb összes költség mellett.

A rendszer költségét tovább csökkenthetjük, ha minimalizáljuk a tényleges anyagmozgatási teljesítményt.

Törekednünk kell a minél jobb helykihasználásra. Az épülethez kapcsolódó költségek az összes raktározási költség akár 40%-át is kitehetik (25% az épület költsége + 15% karbantartási és közüzemi költségek). Emiatt kardinális kérdés a helykihasználás. Törekednünk kell a lehető legmagasabb kihasználtsági arány elérésére. Ilyen példával szolgálnak azok a magasraktárak, ahol maguk a polcsorok alkotják az épület vázát. Így tulajdonképpen a raktárban mindenhol tudunk árut tárolni. Ám ezeknél az épületeknél figyelni kell a helyes súlyelosztásra.

2.1. A raktárak típusai

Különböző módokon definiálhatók a raktárak. Egy megközelítés lehet a fentebb említett fizikai elhelyezkedés szerinti és a rendeltetés szerinti csoportosítás.

A fizikai elhelyezkedés megmutatja, hol található a raktár a vállalat adott telephelyén belül. Ennél a perspektívánál arra törekszünk, hogy a lehető legjobb helyre kerüljenek kialakításra a tárolóegységek. Fontos szem előtt tartani a költségeket és hogy melyik folyamathoz tervezzük építeni. Így magas hasznossági fokkal működhet. Itt kerül előtérbe a pontos tervezés és a támogató infrastruktúra helyes kialakítása. Megfelelő tervezéssel, megfelelő kialakítással, megfelelő raktározási szoftver alkalmazásával kézben tarthatók a kiadások és a szolgáltatás díja. Ha bármelyik nincs kellően kiépítve vagy optimalizálva, az a hasznosság és a gazdaságosság rovására mehet.

Másik szempontunk a rendeltetés szerinti elhelyezkedés. Ez azt mondja meg, hogy hol helyezkedik el a raktárunk az ellátási láncban. Ezek alapján öt típust tudunk megkülönböztetni: alapanyag raktár, félkésztermék-raktár, késztermék-raktár, kereskedelmi raktár, hulladék raktár.

Az én feladatom itt kerül majd előtérbe. A raktárat partícionáljuk, és minden egyes partíciónak megadjuk, milyen cikkcsoportok kerülhetnek bele, ha egy raktárban több különböző árut is tárolunk. Másik eshetőség, ha raktáranként csak egy-egy cikkcsoport tárolható.

Logisztikailag az alapanyag raktár az első a sorban, ugyanis ide érkeznek be a termeléshez szükséges árucikkek, alapanyagok. Egészen a felhasználásukig itt tároljuk őket. Feladatai közé tartozik a megbízható és hatékony anyagtárolás, a rendelkezésre állás, valamint jó hozzáférhetőséget biztosít a soron következő folyamat számára.

A félkésztermék-raktár a gyártási folyamaton belül található meg. Az egyes lépések után van lehetőség a letárolásra, míg a félkész termék a következő fázisba tud lépni.

Készáru-, vagy végtermék raktárak a gyártási folyamat végén találhatóak meg. Ide a terméket már elkészült állapotban tároljuk le, ám még nem szállításra készen. Módosításokat itt már nem végzünk a terméken, mivel kikerült a gyártási folyamatból. Ezen a ponton már csak a kereskedelembe való bekerülését előkészítő lépések történnek. A következő állomás a kereskedelmi raktár.

A kereskedelmi raktárba már szállításra készen, a vevő igényei szerinti állapotban kerülnek a termékek. Ennek a raktárnak a célja nem a hosszú ideig tartó tárolás. A termék csak közvetlenül szállítás előtt kerül be, így itt nem tudnak felhalmozódni. Itt nagy jelentősége van a pontos ütemezésnek és a jó logisztikának.

Az utolsó raktártípus, amiről beszélni szoktak a hulladék raktár. Ide általában a keletkezett selejt kerül. A terméktől függ, hogy ezt a selejtet újra lehet-e hasznosítani, vagy el kell szállítani a megfelelő módon (hulladéklerakóba, veszélyes hulladékot megfelelőképpen kezelni).
Ezen raktártípusok felépítés terén sok mindenben térhetnek el egymástól, hiszen mindegyiket az adott funkcionalitásra hozták létre, hogy a lehető leghatékonyabban működjenek. Felépítésük más, ám helyileg lehetnek egymáshoz közel, akár egy raktárépületen belül is ki lehet őket alakítani. Ezt sok minden befolyásolhatja, de elsődleges tényező a gazdaságosság és a rendelkezésre álló tér. Gyakran a késztermék raktár és a kereskedelmi raktár ugyanaz, nem jelennek meg külön-külön.

2.2. Raktározási funkciók

A raktárak zavartalan működését több funkció hivatott segíteni. Ezek a funkciók zökkenőmentesebbé és ebből következően hatékonyabbá teszik a vállalatot.
Az első ilyen funkció a beérkezés. Ezt további kategóriákra bonthatjuk. Amikor megérkezik az áru, azt fogadni kell, így ezt a részfolyamatot az áru fogadásának nevezzük. Ezt is bonthatjuk kétfelé. Az áru érkezhet a beszállítónktól vagy akár gyáron belül félkész termékként is megérkezhet (visszaérkezhet) a raktárba. Az utóbbi esetben ismertek a paraméterei, tudunk róla konkrétan mindent. Nem úgy, mint az előző esetben, ugyanis itt ellenőrizni kell az érkező szállítmányt, megvizsgáljuk, van-e sérülés rajta. A minőséget több módszerrel is ellenőrizhetjük. Tételesen ellenőrizhetjük az egész szállítmányt. Ez időigényes folyamat, így csak olyan termékek esetén használják, amik ezt megkövetelik. Lehetőségünk van mintavételezésre is. Itt, ha hibát találunk, általánosíthatunk az egész szállítmányra, vagy ezután a vizsgálat után elrendelhetjük a tételes vizsgálatot. Ha hibát találtunk, jegyzőkönyvet és fényképet kell készíteni, mert nem a mi hibánkból sérült az áru. Felvesszük a kapcsolatot a szállítóval a kár biztosító által történő rendezése érdekében. Majd ellenőrizzük a mennyiségét is. A termék minőségi és mennyiségi ellenőrzését, ahol van rá lehetőség, sosem szabad elmulasztani!
Az áru bekerül a raktárunkba. Ha minden rendben van, minden a raktározási folyamatoknak megfelelően történik. Ellenkező esetben, tárgyalni kell a beszállítóval (reklamáció).
Miután megtörtént az átvétel, az áru vagy félkésztermék új fázisba lép. Elkezdődik a betárolási folyamat. A terméket bevisszük a szoftverünkbe, létrehozzuk számára a cikk-kartont. Itt tárolásra kerülnek a termék törzsadatai, mennyisége, fizikai paraméterei. Feladatom megvalósítása után itt további adatot, a termék cikkcsoportját is megadjuk majd a könnyebb tárolás érdekében.

Manapság gyakori eset, hogy a beérkeztetés után az áru nem a raktárba, hanem közvetlenül a termelésbe érkezik (JIT – Just In Time). Ilyenkor nem történik helyfoglalás számára a raktárban. A másik, ehhez hasonló és nagyon elterjedt módszer a JIS (Just in Sequence). Ez azt jelenti, hogy a termékek pont időre érkeznek a felhasználás helyére, beszerelési sorrendben.
A raktározási szoftverek egyre nagyobb térnyerésének köszönhetően, gyakorlatilag azonnal megtudjuk, hova tárolhatjuk, hiszen a szoftver számításaiban rengeteg tényezőt vesz figyelembe. Ezek a tényezők lehetnek a raktár terheltsége, üres helyek elhelyezkedése, az áru prioritási szintje. A gyorsan forgó termékeket általában a polcsorok elejére, alulra helyezik, míg a lassabban forgó termékek hátrább és magasabbra kerülnek a raktárban.
A kapott eredménynek olyannak kell lennie, ami a raktár, a raktározó és a raktáros számára is a optimális. Ha sikeres volt a bevételezés, akkor a készletre vétel ezen a ponton véget is ér, - ahol lehet - a termék megkapja a vonalkódját/QR kódját, ami alapján később beazonosítható lesz majd. A szoftverünkben a termék neve mellett az azonosító kódját és a raktárban elfoglalt helyét tároljuk le, továbbá a korábban említett cikkcsoportokat.
Majd az árut fizikailag is a kijelölt helyére mozgatjuk. Elkezdődik a tárolás.
A következő folyamat a kitárolás. Itt a terméket kivezetjük a szoftverünkből. Feladat, hogy az ellátási lánc (mind a külső és belső) következő folyamata megkapja a neki szükséges mennyiségű és minőségű árut. A fő szempontok ennél a folyamatnál a precizitás (pontosság és gyorsaság együttese).
A kitárolás után szükséges az egyes termékek logisztikai kigyűjtése, ez a folyamat a komissiózás. A komissiózás az áruk megadott megrendelések szerinti kigyűjtése és összeválogatása. Nagy odafigyelést és pontosságot igényel. Ám erre már megvannak az eszközök, mint például a korábban említett raktározási szoftverek, vagy akár az automata – vezető nélküli – targoncák. Megjelentek már automatizált raktárak, de még nem terjedtek el széles körben. Ezekben elkerülhető az emberi beavatkozás, legfeljebb a szállítókra történő rakodáskor érintkezik ember a csomagolt áruval. Így nagy mértékben csökkenthető a hibázás lehetősége.
A komissiózás összetett folyamat, rengeteg résztvevővel. Így fontos a jó információ áramlás, ugyanis sok adatra van szükség a végrehajtásához, de legalább a következő adatokat kell ismernünk, hogy elkezdhessük a folyamatot:

- az áru helye a raktárban,
- megnevezése,
- mennyisége,
- felvétel utáni továbbítás helye,

- mik az áruhiány/kevés áru esetén a teendők,
- a kivétel teljesítése után hol találjuk a listán szereplő következő cikket.

A komissiózást végző alkalmazottnak ezeket az információkat időben és pontosan kell megkapni, hogy tudja teljesíteni a feladatát. Ha ez nem történik meg fennakadások adódhatnak a raktár körforgásában. Szoftver vezérelt kitárolással és azonosítással (vonalkód, QR kód, raktári polc kódja) minimalizálható a tévedések száma. A folyamat maga pedig jelentősen gyorsítható.

A komissiózástól nagy mértékben függenek a soron következő folyamatok. Különösen a mai vevőközpontú piacon, itt érezhető legjobban a vevő és a vezetőség felől érkező nyomás. A rendelések kielégítésének érdekében itt terméktől függően készlettöbbletet halmozunk fel, hogy egy esetleges fennakadás esetén is zavartalan legyen a vevő felé a szállítás. Ez ugyanúgy igaz, ha gyártási folyamatok közötti alapanyag utánpótlásról van szó.

A komissiózást még tovább bonthatjuk egylépcsős és kétlépcsős komissiózásra, amik szintén még tovább specifikálhatók. Ezeket ismertetem a továbbiakban.

Egylépcsős komissiózás alatt értjük azt, amikor a gyűjtő számára jól hozzáférhető helyen kerülnek elhelyezésre az áruk a tárolóhelyen. Így a gyűjtő össze tudja szedni a kellő mennyiséget és továbbítja a következő folyamatnak.

Ha kétlépcsős komissiózásról beszélünk, nem történik közvetlen gyűjtés. Először a használt raktározási séma alapján készletre került árukat típusok szerint kigyűjtik, majd átcsoportosítják egy olyan rakodó/tároló területre, ahol egységrakományokat képeznek belőlük.

Az egységrakomány képzés azt jelenti, hogy a legtöbbször azonos típusú árukat nagyobb rakományokká állítjuk, csoportosítjuk össze, melyek könnyebben kezelhetők, tárolhatók, mozgathatók. Ha ezeket az egységrakományokat megfelelően alakítjuk ki, rengeteg előnyre tehetünk szert velük:

- az egyszerre mozgatható árutömeg jelentősen nő, így kevesebb mozgatással helyezhetjük át a terméket,
  - kevesebb kapacitást igényel, és csökkennek a költségeink,
- kisebb helyen tárolható ugyanaz a mennyiség, jobban kihasználható a raktár hasznos területe,
- könnyebb megfelelő mozgató eszközt találni az áthelyezésre,
- gyorsulnak a load folyamatok,
- csökkenthető az esetleges balesetek száma és ezzel a károk is.

Ezért is fontos a pontos cikkcsoportba rendelés. Ha jól sikerült a cikkcsoportba rendelés, könnyebben kialakíthatók az egységrakományok, könnyebben kezelhető az áru a raktározási folyamatok alatt.

Az egy- valamint kétlépcsős komissiózás alfolyamatait lehetőségünk van több módon jellemezni, osztályozni. A komissiózás első lépése az áru elkészítése. Ez történhet statikus vagy dinamikus módon. Ez az osztályozás a termék szempontjából történik. Statikus elkészítésnél a gyűjtőt mozgatjuk az áru irányába, míg a dinamikusnál a terméket mozgatjuk a gyűjtőhöz.

A kiszedés folyamata két tengely mentén történhet. Ha csak X tengely mentén, horizontálisan mozgatjuk a terméket – a talajszinten – akkor egydimenziós kiszedésről beszélhetünk. Ebben az esetben a kiszedés maga történhet kézzel, mozgatóeszköz, vagy berendezés segítségével. Ha X és Y tengelyek mentén, horizontálisan és vertikálisan is mozgatjuk egy emelő segítségével, akkor kétdimenziós kiszedésről van szó. Ennél az esetnél a gépi mozgatás a legjellemzőbb.

Végül pedig az áruleadás következik. Ez is történhet centralizált és decentralizált módon. Jelentősége nagy, hiszen kihat a rendszerünk hatékonyságára, erőforrásaink kihasználtságára, valamint ezekből adódóan a költségeinkre is.

A centralizált mód azt jelenti, hogy termékeinket (készleteinket) egy helyen, egy nagy kiterjedésű tároló egységben (egy nagy raktárban) helyezzük el. A kiszedés befejezésével az árukat egy előre kialakított központi helyre szállítjuk. Emellett több pozitívumot is fel tudunk sorakoztatni. Tekintsük át ezeket.

A centralizálás előnyei:

- készleteinket sokkal jobban ellenőrzésünk alatt tudjuk tartani,
  - készletszintekre és készlethelyzetekre nagyobb rálátás,
- jobb erőforrás kihasználás lehetséges,
  - mind emberi, mind terület, mind gépi erőforrások,
    - könnyebb koordináció,
- minőség-ellenőrzés elvégzése egyszerűbb,
- értékes vagy veszélyes árukat jobban ellenőrzés alá lehet vonni.

Elég komoly érveket tudunk a centralizált megoldás mellé felsorakoztatni.

A decentralizált megoldást az hívhatja életre, hogy:

- több terméket kezelünk (alapanyag, félkész termék stb.),
- több raktárépületet igénylő mennyiségű árut kezelünk,

- azt szeretnénk, hogy a termék kevés időt töltsön a rendszerünkben (rövid átfutási idő),
- a különböző termékek más-más kezelést, raktározási módot igényelnek.

Természetesen emellett a megoldás mellett is tudunk érveket felsorakoztatni.

- felhasználási helyhez közel tudjuk tárolni az árut,
  - kevesebbet kell mozgatnunk a terméket a felhasználó folyamat és a tároló között,
    - kisebb az esély az anyagi kárra,
- gyors utánpótlási lehetőség (szinte azonnal a rendelkezésünkre áll a készlet),
- a felhasználó jobban rálát a termékeire, így több információval rendelkezik róla, jobban informált,
  - emiatt nagyobb bizalom épül ki a felhasználó és a raktározó között.

Nevesítsük, mit is takar a decentralizált megoldás. Ekkor a kiszedett árukat egy anyagmozgóval közvetlen a kiszedés után mozgatjuk a kijelölt gyűjtő helyre.
A különböző alfolyamatok megfelelő típusaiból összeállítható a vállalatunk számára a leghasznosabb, leggazdaságosabb megoldás a költséghatékonyság és a nagyobb hasznossági fok érdekében.
Vizsgálhatjuk még, hogy hol érdemes a komissiózást végrehajtani: tárterületen belül, avagy kívül. Mindegyiknek megvannak a maga előnyei és hátrányai. A végső döntést – hogy melyiket is használjuk – a vállalatunk és a lehetőségeink határozzák meg.
A következőkben a fenti két eset előnyeit és hátrányait vizsgálom majd.
Elsőre vizsgáljuk a tárterületen belüli elhelyezést. Kezdjük az előnyeivel:

- könnyen ki lehet alakítani az egylépcsős komissiózást,
  - az áru majdnem a kitárolási területen található,
  - a komissiózás egyszerűen elvégezhető az áru kitárolásakor,
- a szállítás nem vesz sok időt igénybe, egyszerűbb,
- könnyebb az információ áramlás.

Hátrányai:

- nagy helyet foglalhat el a rakodóterületen, így egyszerre csak egy komissiózás végezhető, ez a kihasználtság tekintetében nem kedvező,
- kisebb mennyiségű áru halmozható fel, a véges rakodótér miatt.

Tárterületen kívüli komissiózás előnyei:

- nem foglalunk el teret a raktárban, mivel kimondottan rakodási célú területet használunk,
  - nagyobb mennyiségű árut halmozhatunk fel,
    - jobb a kihasználtság, hatékonyabb a raktár,
- raktáron kívül foglalnak el területet, így a raktárak körül több ilyen pont is létesíthető,
  - egyszerűbb, jobb, hatékonyabb a kialakítás és működés,
  - gyorsabbak lehetnek a load folyamatok.

Hátrányai:

- mivel nem a raktárban találhatóak, kétlépcsős komissiózást igényel,
  - a kétlépcsős komissiózás több szervezést, nagyobb energiabefektetést kíván,
  - nőhetnek a raktáron belüli szállítási idők,
- előfordulhat, hogy az árut a raktár távoli pontjáról kell átmozgatni a rakodási területre,
  - hosszabb távon mozgatjuk az árut, nagyobb az esély a sérülésre.

A komissiózást követő zárófolyamat a kiszállítás. Ezt sok esetben nem a raktározással foglalkozó vállalt intézi, hanem egy harmadik fél, egy logisztikai vagy fuvarozó vállalat.
A komissiózással még nem ér véget a raktár feladata, ugyanis a szállító (kamion, vasúti kocsi, hajó) eszközt meg is kell rakni. Ez az a folyamat, amit még nem sikerült teljesen automatizálni, itt még szükség van emberi beavatkozásra. Ám a folyamatot nagy ütemben gyorsíthatja a jó szervezés és csapatmunka. További segítség lehet, ha a komissiózó és rakodó területeket elválasztjuk egymástól, hogy a két folyamat szegmentáltan tudjon működni. Ezt a megoldást különösen nagyobb szétosztó depókban szokták alkalmazni, ahol a fő profil az esetenkénti, sürgős megrendelések teljesítése. Ebből adódik, hogy kisebb létszámú személyzetet alkalmazhatunk a kiszedési folyamathoz, mert viszonylagosan kis területet kell bejárni a munkásoknak és a szükséges raktérkészlethez egyszerűen hozzáfér. A komissiózást végző alkalmazottaknak csak a tároló területig kell elmenniük, nem kell esetlegesen az egész raktárat bejárni.
Így a komissiózás gyorsul, a rendszerben (raktárban) töltött idő is csökken. Ebből következően nőhet a forgalom, ugyanis rövidebb egy kérés teljesítésének ideje. A készleteket szükség esetén a tartalék tárolóról lehet összegyűjteni. A tartalék tárolón tárolt áruk összetételét akár informatikai eszközökkel (öntanuló algoritmusok) is támogathatjuk, ezzel gyorsítva a

folyamatot. Más esetekben nem szükséges ily módon szétválasztani a raktárat, a fő tároló területet egységesen tarthatjuk.
Ezeket az esetek:

- alacsony forgalom,
- homogén árukészlet,
- komissiózás nem szükséges.

A komissiózásnak is megvannak a maga eszközei. Vegyük sorra ezeket is. Elsőnek nézzük a talán legegyszerűbb módszert, a kézi komissiózást. Ez egy lassú folyamat, maximális gyűjtési magassága van (maximum 2-3 méter), meglehetősen időigényes. A kézi kocsi nyújtja a legnagyobb támogatást ennek a megoldásnak.
A raktárakról az emberek jó részének a targoncát vezető raktáros jut az eszébe. Meglehetősen elterjedt módszer. Sok helyen nem lehetséges a teljes automatizálás, így ez a legkézenfekvőbb módszer. Megjelentek már automatizált targoncák és teljesen automatizált, szenzor és szoftver vezérelt raktárak is. Ebben az esetben már nincs szükség közvetlenül emberi beavatkozásra. Csak egy informatikusra, aki a gépteremből figyeli, hogy minden a maga medrében folyik-e.
A jól optimalizált kigyűjtő rendszerben elkerülhetők a torlódások. Az optimalizálás a kellő számú gép és eszköz számának meghatározását jelenti. Ezt a raktárunk paraméterei és a forgalom is befolyásolja.
Ezzel ez a raktározási folyamat véget ér, majd mikor az áru megérkezik céljához, újra bekerül egy raktározási ciklusba, mígnem a fogyasztóhoz kerül.
A sikeres raktározáshoz a fent említett funkciók pontos megtervezése és jól összehangolt működése megkerülhetetlen. Valamint nem utolsó sorban...
Számos dolog szükségeltetik a sikeres működéshez:

- megfelelő digitalizáltság,
  - szükséges, up-to-date szoftverek használata,
- megfelelő kommunikáció,
  - mind a raktáron belül, mind a vevővel és szállítóinkkal,
- raktározási eszközök (emelők, számítógépek stb.) rendszeres karbantartása.

Ha ezeket mind sikerült szolgálatunkba állítani, rengeteg előnyünk származhat majd belőle:

- az áru rövidebb időt tölt raktárunkban,
- kevesebb atrocitás, sérülés érheti,
- vevőink elégedettsége nő,
- költséghatékony működés,
  - magasabb profit.

2.3. Áramlatok

Az áramlatok célja, hogy kijelölje „a járást” a raktáron belül. Létjogosultsága hatalmas, hiszen a raktárkezelő szoftverünknek a robotok mozgatásánál, a rakodóterületek kialakításánál óriási szükség van rájuk.

Tervezésüket algoritmusokkal lehet a legegyszerűbben megoldani. Ezek az algoritmusok végig járják az összes lehetséges utat a raktárban, mindaddig amíg megtalálják a legcélszerűbbet, a rendszerünk számára optimálisat.

Feladata a folytonos anyagáramlás ellenőrzése, fenntartása és biztosítása, ezenfelül áramlatok nem keresztezhetik egymás nyomvonalát, továbbá kerülniük kell a zsúfolt, forgalmas részeket.

Az áramlatok monitorozásával ismerjük gépeink és termékeink mozgását, helyét a rendszerünkben. Jól megtervezett áramlatokkal gyorsíthatók a raktározási folyamatok.

Kétféle áramlat alkalmazása a legelterjedtebb. Az „U” áramlat és az egyenes áramlat.

Az „U” áramlatról már korábban beszéltünk. Ez jelenti azt, hogy a termékeket prioritás szerint helyezzük el a raktárakban. Akkor célszerű emellett döntenünk, ha a fogadó és kiadó létesítmény közös. Alkalmazása számos előnnyel jár:

- a rendelkezésre álló területet és berendezéseinket jobban ki tudjuk használni,
- kisebb helyigénnyel rendelkezik,
  - ahhoz viszonyítva, mintha külön lenne a fogadó és kiadó terület,
- magasabb biztonsági fok érhető el,
- egyszerűbb kapcsolatkiépítés a külső környezettel.

Párja az egyenes áramlat. Létesítésének feltétele, hogy a fogadó és kiadó terület külön, az épület két végén helyezkedjen el.

Ez a megoldás rugalmatlan lehet. Időigényes, ugyanis minden árunak végig kell menni a teljes épületen. Emiatt nehezebb ellenőrizni, emellett nehezebben bővíthető.

Bár vannak esetek mikor használata célszerű lehet:

- ha a be- és kiszállító járművek mérete, típusa, jellege ezt indokolja,
- ha nem szeretnénk, hogy a be- és kiáramló áruk összekeveredjenek és ezt szeretnénk megelőzni,
- ha szükséges, hogy a raktározás a termelési folyamat része legyen,
- ha a készáru raktárat az üzemünk végén létesítettük.

2.4. Megvalósítás NAV 2016 környezetben

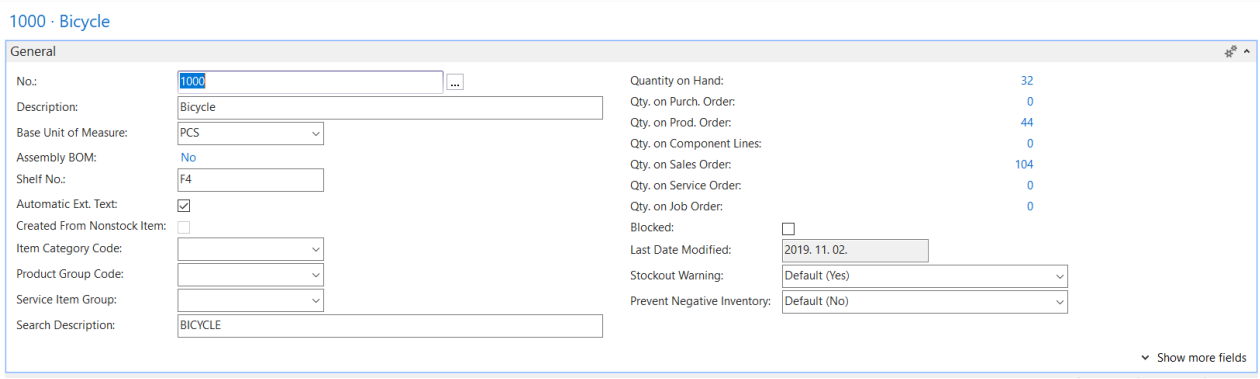
Egy ERP rendszernek tartalmaznia kell a raktározási folyamatok megvalósítását is. Így összegezni tudja azokat a rendeléseket, melyeknek teljesítési dátuma egy adott nap, vagy néhány napon belül esedékessé válik. Ehhez természetesen szükséges, hogy az adott termék a megfelelő mennyiségben és minőségben fellelhető a raktárunkban.

Ha megtörtént a kivételezés, azt az ERP rendszerben is jeleznünk kell. Ez adatok frissítésével történik meg. Frissíteni kell az adatokat a főkönyvi könyvelésben, hogy a raktárban lévő termék értéke csökkent. A logisztikai alrendszerben is jelezni kell, hogy a termékből rendelkezésre álló mennyiség csökkent. Ezek után pedig kiállíthatjuk a számlát.

Továbbá kiállítható a szállítólevél is.

A NAV a kivételezést és a rendszer adatainak frissességét is könyvelési folyamatokon keresztül biztosítja, ezért mondhatjuk, hogy erőteljesen könyvelés alapú rendszer.

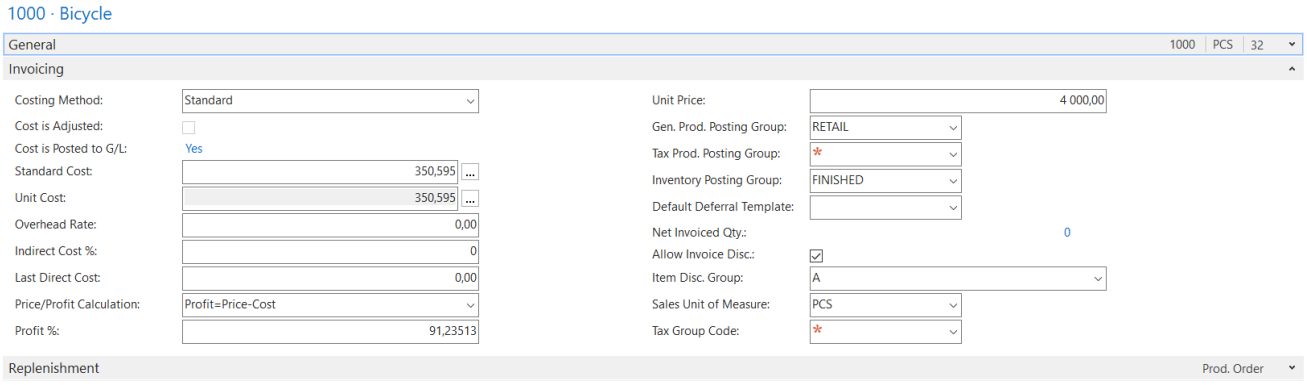
A cikkek listáját több különböző úton is tudjuk érni. Erre egy példa: *Departments* ⇒ *Purchase* ⇒ *Planning* ⇒ *Lists* ⇒ *Items*. Itt egy tetszőleges cikken duplán kattintva tudjuk megtekinteni a cikk-kartont. A cikk-kartonon minden, a cikk-kel kapcsolatos információt megtalálunk.



6. ábra: Cikk-karton General [Általános] fül

A General [Általános] találjuk a főbb alap adatokat. Ezek közül a fontosabb mezők:

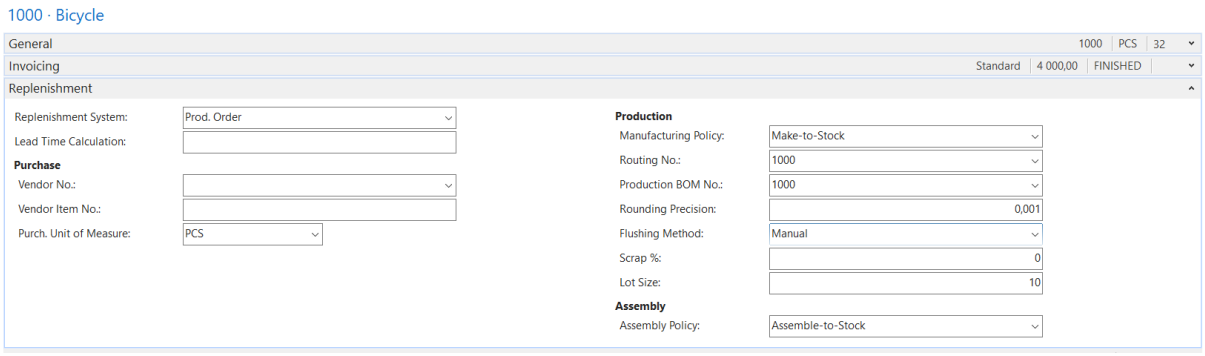
- **No.:** cikkszám (automatikusan generált/betöltött/kézzel beírt)
- **Description:** a cikk megnevezése. Ennek megadása nem kötelező, mivel a cikk a No. alapján már azonosítható. Megnevezés megadása logikailag kötelező, célszerű.
- **Base Unit Of Measure:** a cikk alapegysége (PCS - darab, Kg, Liter, stb…)
- **Item és Product Group Code:** a cikket ez alapján tudjuk besorolni. A besorolás alapja valamilyen közös tulajdonság.
- **Quantity on Hand:** a termékből raktáron lévő mennyiség
- **Qty. on Purch. Order:** ennyit rendeltünk meg a cikkből



7. ábra: Cikk-karton Invoicing [Számlázás] fül

A számlázás fülön is sok hasznos információ nyerhető ki. Ezek közül a fontosabbak:

- **Costing Method:** a készletértékesítés módját állíthatjuk itt be (pl.: LIFO, FIFO)
- **Unit Cost:** a cikk közvetlen költsége (egységköltség)
- **Indirect Cost %:** közvetett költségek összege
- **Price/Profit Calc.:** az ár/haszon kiszámításának módja, több lehetőség közül választhatunk, e beállítás alapján számolódhat ki az egységár
- **Profit:** a haszon mértéke
- **Tax Prod. Posting Group:** Termék-, Áfa-, Készletkönyvelési csoportot állítható itt be
- **Item Disc. Group:** beállítható, hogy a cikk mely engedménycsoportba tartozik
- **Sales Unit of Measure:** eladási mennyiség, akkor lesz fontos ez a beállítás, ha a terméket más egységben adjuk el, mint ahogyan beszereztük

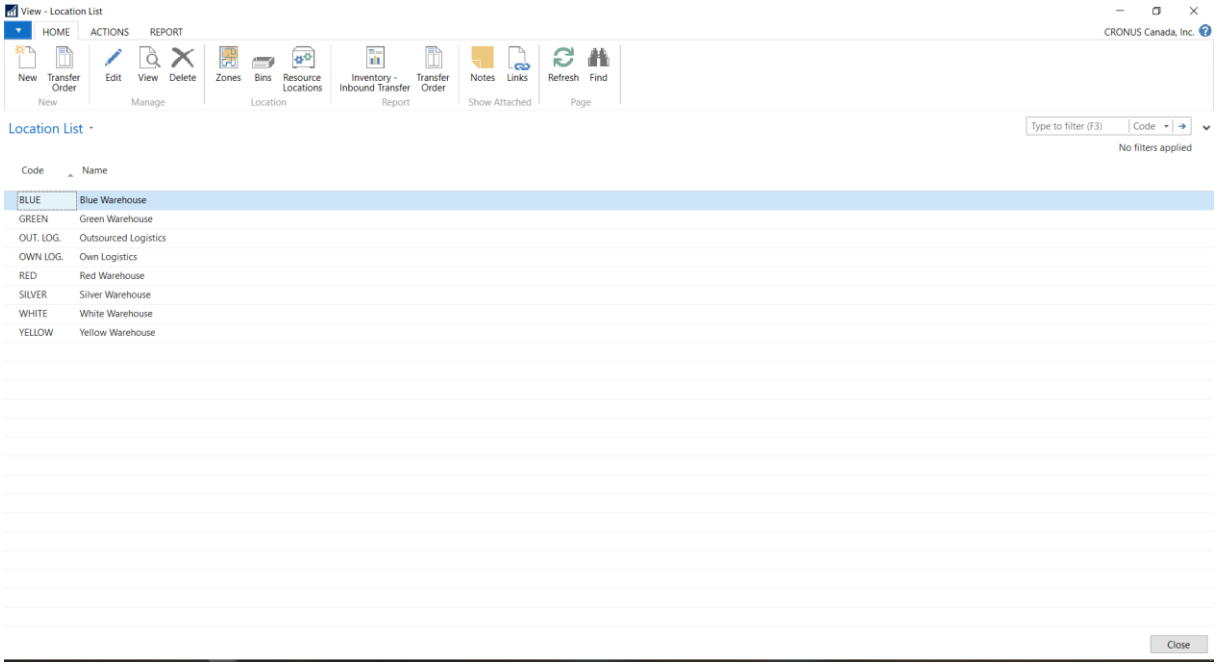


8. ábra: Cikk-karton Replenishment [Feltöltés] fül

A következő fülön a feltöltéssel kapcsolatos információk állíthatók. A fontosabb mezők ezen az oldalon:

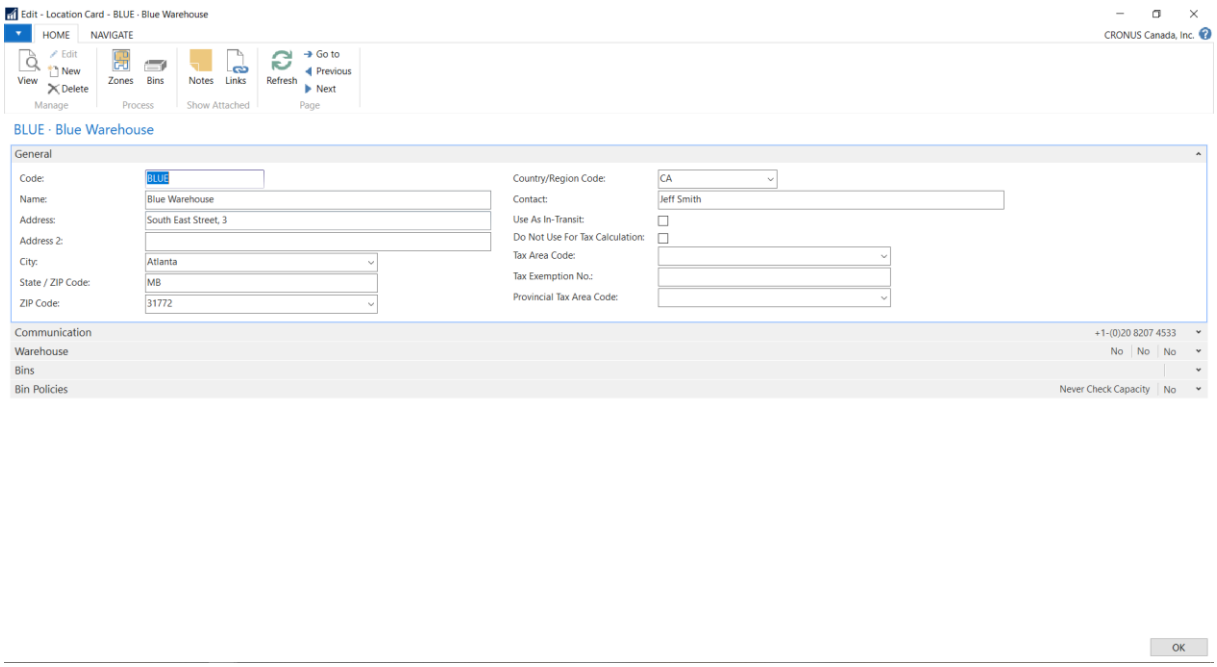
- **Replenishment System:** itt adjuk meg, hogy a termékkészlet milyen módon kerül feltöltésre (pl.: gyártjuk, beszerezzük)
- **Vendor No.:** beszerzés esetén ezzel a számmal tudjuk azonosítani a szállítót, akitől a terméket (alapértelmezetten) rendeljük
- **Purch. Unit of Measure:** beszerzési mértékegység

A következőkben vizsgáljuk meg a raktárakat (Warehouse) is. A rendszerünkben a következő elérési úton találhatjuk meg raktárainkat: *Departments* ⇒ *Warehouse* ⇒ *Setup* ⇒ *Locations.* A cikkekhez hasonlóan itt tudunk újat felvenni, vagy a meglévőket módosítani.



9. ábra: Locations List

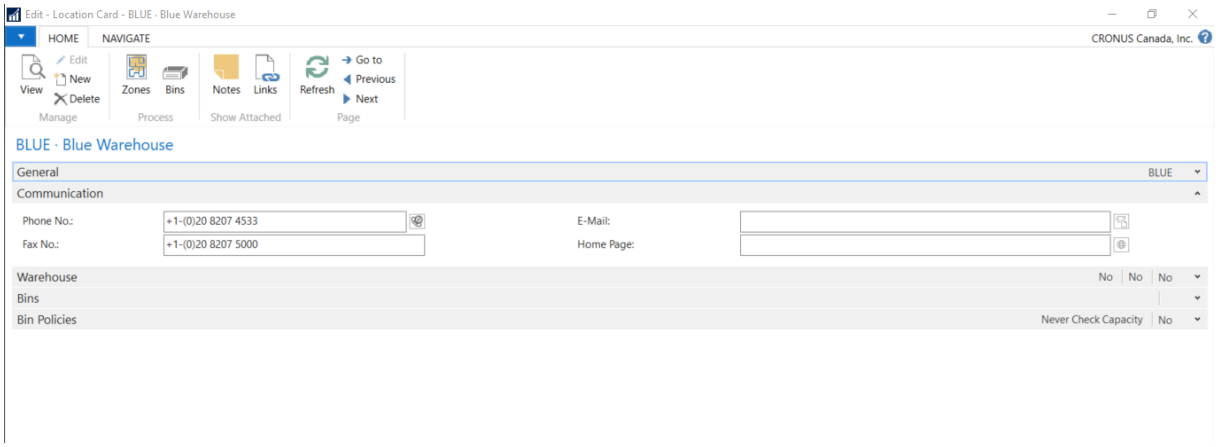
Példaként „nézzünk szét” a Kék raktárban (Blue Warehouse).



10. ábra: Blue Warehouse [Kék raktár] – General [Általános] fül

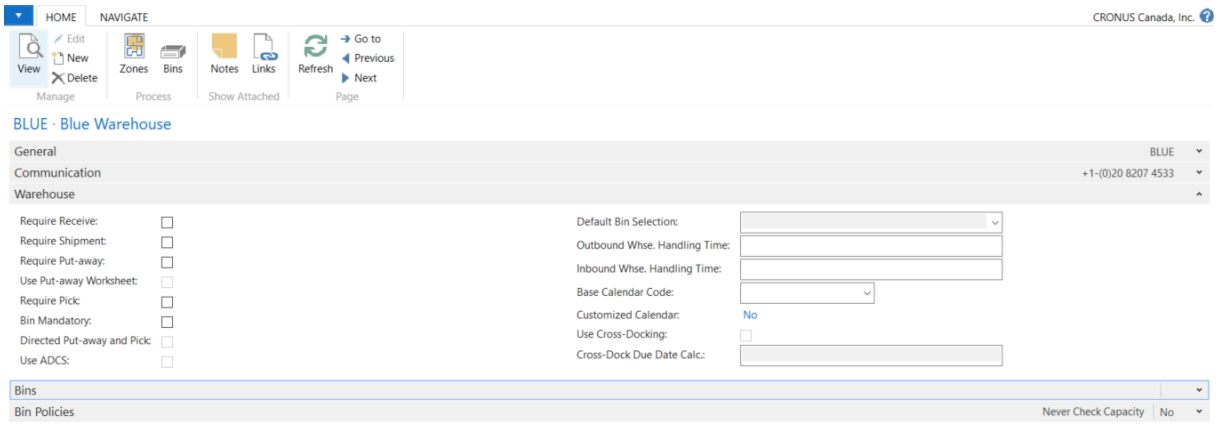
A raktárkartonon az Általános fülön a raktár törzsadatait találjuk meg:

- **Code:** raktár kódja
- **Name:** raktár neve
- **Address:** raktár címe
- **City:** mely városban van a raktár
- **State/ZIP Code:** az állam neve
- **ZIP Code:** irányító szám
- **Country:** ország
- **Contact:** ügyfél neve



11. ábra: Communication [Kapcsolat] fül

Ezen a fülön a raktár telefonos és faxos elérhetőségét, valamint az e-mail címét adhatjuk meg. Ezeken felül megadható egy webcím is, ami a raktár honlapjára mutat.
A következőkben mélyebben tekintjük át a raktárat, a Warehouse [Raktár] fül vizsgálatával.



12. ábra: Warehouse [Raktár] fül

Az itt megtalálható attribútumok jó része BOOLEAN változó, értéküket a jelölő négyzetekkel változtathatjuk. Ezek a változók apróbb módosításokat, további specifikálásokat tesznek lehetővé:

- **Require Receive:** beérkezés kötelező
- **Require Shipment:** kiszállítás kötelező
- **Require Put-away:** eltárolás kötelező
- **Require Pick:** kötelező kiszedés
- **Bin Mandatory:** kötelező raktárhely
- **Use Cross-Docking:** tovább szállítás

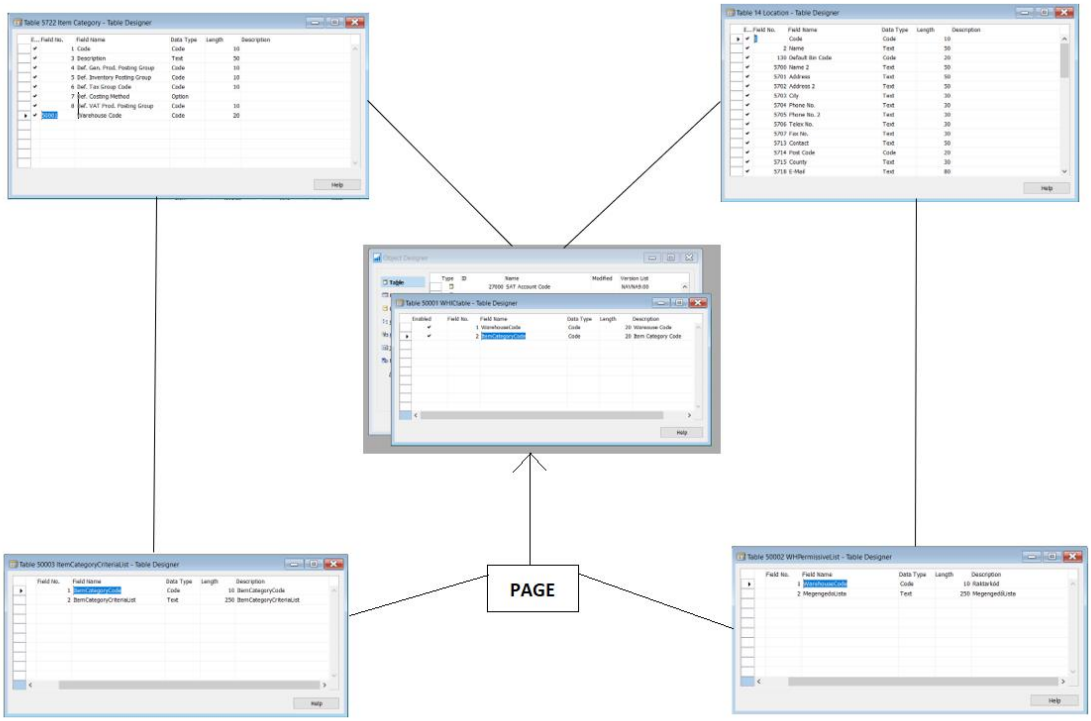
Megadható még ezen az oldalon, hogy milyen naptár kezeléssel működjön a raktár (**Base Calendar Code** – GB vagy Service).
A következő menüben (Bins - Raktárhelyek) a raktárhelyek beállításait módosíthatjuk/adhatjuk meg. Az utolsóban (Bin Policies – Raktárhely elve) pedig a raktárhelyekre vonatkozó elveket állíthatjuk be.
Most, hogy ismerjük a cikk- és raktárkarton felépítését, attribútumait le lehet vonni a következtetéseket, elkezdődhet a megvalósítás dokumentálása.

3. Témalabor és Önálló Laboratórium tárgyak konklúziója

A Témalabor tárgyam keretei között ismerkedtem meg teljes mértékben a Cosmoval. A Témalaborral egyidőben tartottak egy órát, mely végleg eldöntötte, hogy ERP rendszerekkel szeretnék foglalkozni. Az órának köszönhetően jobban beleláttam a NAV működésébe. Megismertem a rendszer működését, a vállalati folyamatokat és a vállalati policy-ket.

Az ERP rendszerek jelentősége és létjogosultsága egyre nő a mai üzleti életben. Az ERP rendszerek a vállalat belső folyamatait ölelik át. Ezek segítségével hatékonyabban tervezhetők az erőforrások, könnyebben koordinálhatók a raktári mozgások és a könyvelési, riportolási folyamatok. Az ERP rövidítése feloldva is erre utal. ERP = Enterprise Resource Planning, azaz vállalati erőforrás tervező. Az ERP rendszerek zászlóshajója a NAV és jelenleg forgalomba kerülő, up-to-date rendszernek számító Business Central. Azért esett a NAV 2016-os verzióra a választásom, mert oktatási keretek között ehhez kaptunk licence-et, így evidens volt a használata.

Témalabor dolgozatomban megismertem a NAV tábla objektumait, megértettem az adatbázis rendszerének működését. Így létre tudtam hozni egy mostanra kezdetlegesnek mondható adatbázis módosítást.

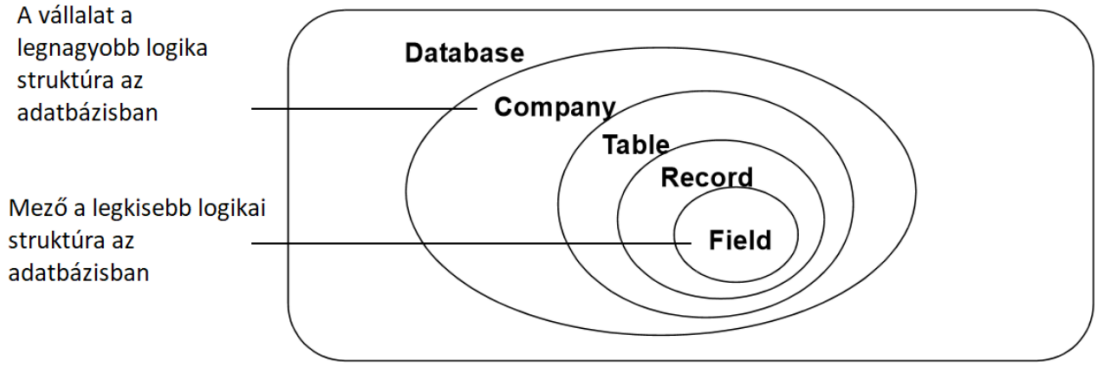


13. ábra: Kezdetleges adatbázis módosítás

Önálló Laboratórium dolgozatom keretében már a NAV további objektumaival ismerkedtem. Elkezdtem a Page Objektumok és a Codeunit létrehozását. A dolgozat előre haladtával látszott, hogy nem megfelelő az adatbázis kiegészítésem, így ezt át kellett alakítanom. Szakdolgozat keretein belül pedig még további módosításokra volt szükségem.
Page és CodeUnit objektumokat hoztam létre Önálló Labor dolgozatban. Ezekkel a fejlesztésekkel egyre jobban átláttam a rendszer működését. A Cosmotól kapott NAV fejlesztési dokumentumok, az órai anyag és a VIR2 óra anyagai is segítettek az objektumok helyes létrehozásában, a fejlesztési logika megértésében.

3.1 C/AL nyelv

A NAV C/AL vagy AL nyelven íródott. Ez egy Pascal alapú nyelv, mely objektum alapú, de nem objektum orientált nyelv. Arra lett fejlesztve, hogy hatékonyan lehessen vele adatbázis manipulációt végrehajtani. Érdekessége, hogy minden objektumában kódolhatunk. Az előző fejezetben bemutatott fejlesztői környezetben (3. ábra) választhatunk a lehetséges objektumaink közül. Ezek pedig a már korábban említett: Table, Page, Report, Codeunit, Query, XMLport, MenuSuite. A Table adatok tárolására szolgál. A Page (korábban Form) adatokat jelenít meg a felhasználó számára. NAV 2009 óta találjuk meg a Page objektumokat a rendszerben. A Reporton az adatokat szűrve láthatjuk, papíralapú kimutatások alapját képezi. A Codeunitban függvényeket tárolunk, az OO programozásban az osztály fogalmával feleltethető meg. A Query SQL nyelve optimalizált, segítségével kérhetünk le adatokat egyszerre több táblából, megadhatjuk, ezek a táblák mi alapján legyenek összekapcsolva, majd az eredményre szűrőket állíthatunk be. NAV 2013 óta van jelen a rendszerben. Az XMLport segítségével adatokat importálhatunk és exportálhatunk az adatbázisunkba, a nevéből adódóan XML alapon. A MenuSuite-ban a megjelenítendő objektumokat találjuk. Itt kell majd a saját kiegészítőmet is „kitenni” a menübe.



14. ábra: Az adatbázis logikai struktúrája

Természetesen ebben a nyelvben is vannak egyszerű (Integer, Decimal, Text, Date stb) és összetett típusok (Record, Page, Codeunit, Option), ám ez utóbbi eltér a más nyelvben megszokottaktól. Ugyanis itt az összetett típusok alatt egész NAV objektumokat értünk. Record típusú változóként egy egész táblát adhatunk át, melyre utána előre definiált függvényekkel szűrhetünk, iterálhatunk rajtuk és módosításokat is végezhetünk. A következőképpen hivatkozhatunk rá: TáblaNeve.MezőNeve/FüggvényNeve. Mint minden objektumot, Codeunitot is átadhatunk változóként. Ilyenkor hozzáférünk az összes benne megírt függvényhez. Hivatkozása is ugyanúgy működik: CodeunitNeve.FüggvényNeve.
Többféleképpen történhet a deklarálás C/AL-ben. Megkülönböztetünk egyszerű értékadást és triggeren keresztül történő értékadást.
Egyszerű értékadás: valtozoneve := ertek;
Triggeren keresztül: Tablareferencia.VALIDATE(Mezoreferencia, ertek);
Mint minden más nyelvben, itt is jelen vannak az elöl tesztelő és hátul tesztelő ciklus, az iteráló ciklusok (for, foreach) és a szekvenciák is. De a leggyakrabban használt ciklus a rendszer logikájából fakadóan a REPEAT – UNTIL.

3.2. Összegzés

A fejezetben bemutattam, miképp fogtam hozzá korábbi önálló munkáimhoz, melyek szakdolgozatom alapjául szolgáltak. A Vállalatirányítási rendszerek főmodul minden tárgya segítségemre volt. Mindegyik tárgyból hallottam, tanultam olyat, amelyet fel tudtam használni jelen dolgozat elkészítéséhez. Mivel korábbi önálló munkáim egymásra épülnek, a korábbi tapasztalatok alapján készíthettem el a végleges kiegészítőt. A választott alkalmazás verzió sem okozott hátrányt, régebbi szoftverről lévén szó. Mivel nem cégspecifikus alkalmazásról beszélünk, bármelyik cég rendszerébe igény szerint implementálható, akár újabb, akár régebbi típusú Navisionnel rendelkezik.
Ismertettem a C/AL nyelvet és annak sajátosságait. Mivel órai keretek között csak JAVA nyelven programoztam, nehéz volt hozzászokni a nyelvi elemekhez és működésükhöz. Önálló munkáim végére átláttam a rendszerben nagyon gyakori fej-soros (Page-SubPage) kapcsolatok működését. Ennek helyes összeállításával és az összehasonlító algoritmus megírásával töltöttem el a legtöbb időt. Az elején sok hibára futottam, ám az interneten található dokumentációban minden hibámra találtam megoldást. A NAV-hoz tartozó dokumentáció teljeskörű, példákkal szemléltetett, így munkám során is sokszor fordulok hozzá segítségért.
A fent említett kapcsolatokon kívül beleláttam a NAV adatbázis struktúrájába, amit a nyelvi elemek és sajátosságok alapján sikerült megértenem.

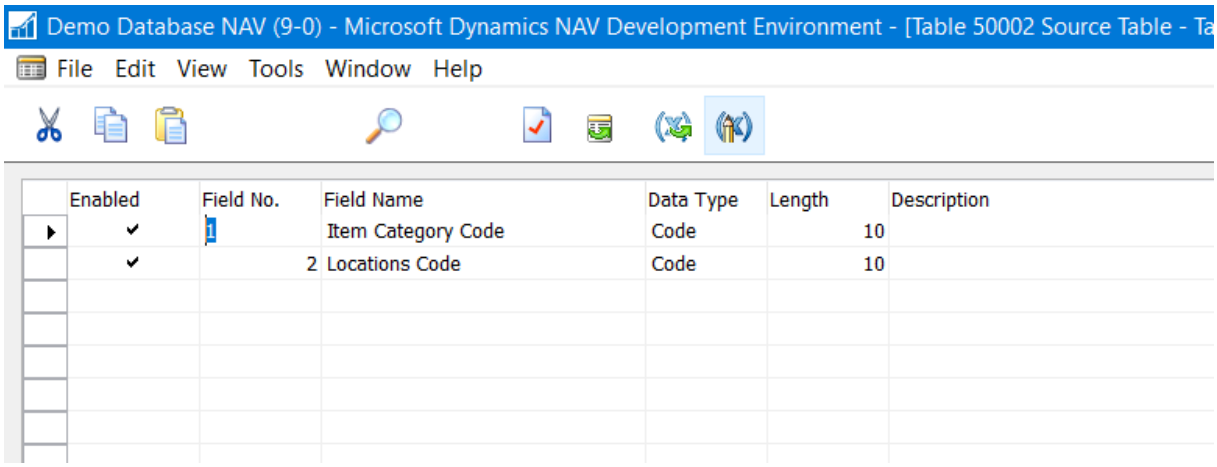
A következő, 4. fejezetben mutatom be a feature létrehozásának folyamatát. Az egyes lépéseket a kliensről vagy a fejlesztői környezetről készített screenshotok segítségével színesítem.

4. A feature létrehozásának folyamata

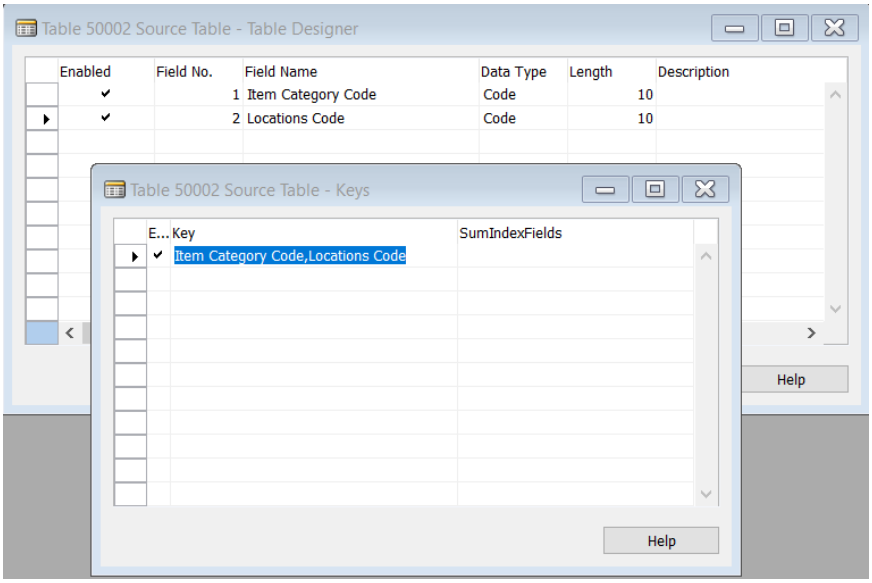
4.1. Eddigi megvalósítás összefoglalása, kiegészítése

A Témalabor és Önálló Labor tárgyak keretében elkezdett feladatot folytatva némileg újra gondoltam a megoldásomat. Más elnevezéseket vezettem be, a könnyebb megértés érdekében. A Location Parameters és az Item Category Parameters táblákban a Parameter text típusát lecseréltem Option típusra. Szakdolgozat keretében pedig Code Típusúra. Ez alapján kapcsolódik össze a Parameter Table-el. Erre azért volt szükség, hogy dinamikusabban legyen bővíthető a feature. Továbbá könnyebb lesz vele a munka a későbbiekben.
Az összehasonlítást és a letárolást a NAV CU-k segítségével valósítom meg.
Az Object Designerben létrehoztam két Page objektumot. 50000-es sorszámmal ([Page 50000 Linking Page]), forrástáblája (SourceTable Property) az Item Category. A másikat LinkingPageList-nek neveztem el, ennek az Item Category a forrás táblája.
Az adatbázisban még további módosítás a Temporary Location ([Table 50005 Temporary Location]). Ez a tábla ugyanúgy néz ki, mint a Source Table, attribútumai és kulcsai a Location Code és az Item Category Code. Amikor lefut az összehasonlítás, a Management CodeUnit Compare függvénye ide írja majd be a szűrt adatokat, és a Location ParameterList Subform ezt használja a megjelenítéshez.
Az utolsó létrehozott tábla a Parameter Table. Ebben a táblában egy Parameter ID és egy Parameter Name mező található. A tábla kulcsa két elemű, az ID és a Name is a kulcs részét képezik. A táblát fel is töltöttem a korábbi OptionString Értékekkel. Az ID-k sorszámozása jelenleg 1-től 8-ig, a nevek pedig: Cold Storage, Unheated, Tempered, Loading Ramp, Racking, Arrays Storage, Chemical, Bulk Storage.
A Linking Page-en létrehoztam két Groupot a Page Designerben, egyikben az Item Category kódját és leírását jelenítem meg a szűrt Location Code-ok mellett, míg a másikban (FactBox) az aktuális Item Category paramétereit. Először nézzük az első Groupot.
A Code és Description attribútumokat jelenítem meg egy-egy Field-ben. Mivel az oldalon egymásba ágyazott page-ekre van szükség, ezért létrehoztam egy Subpage-et az Object Designerből, amin az adott cikkcsoporthoz szűrt Location kódját láthatja majd a felhasználó. A Subpage neve Location ParameterList, típusa Listpart. Az oldal forrás táblája a Temporary Location, a tábla mindkét attribútumát rátettem a Page-re, viszont az Location Code attribútumot jelenítem csak meg, mert csak ez az adat szolgál többletinformációval a felhasználónak, továbbá mindennemű félreértés és hiba elkerülése végett nem célszerű megjeleníteni az Item Category Code-ot.

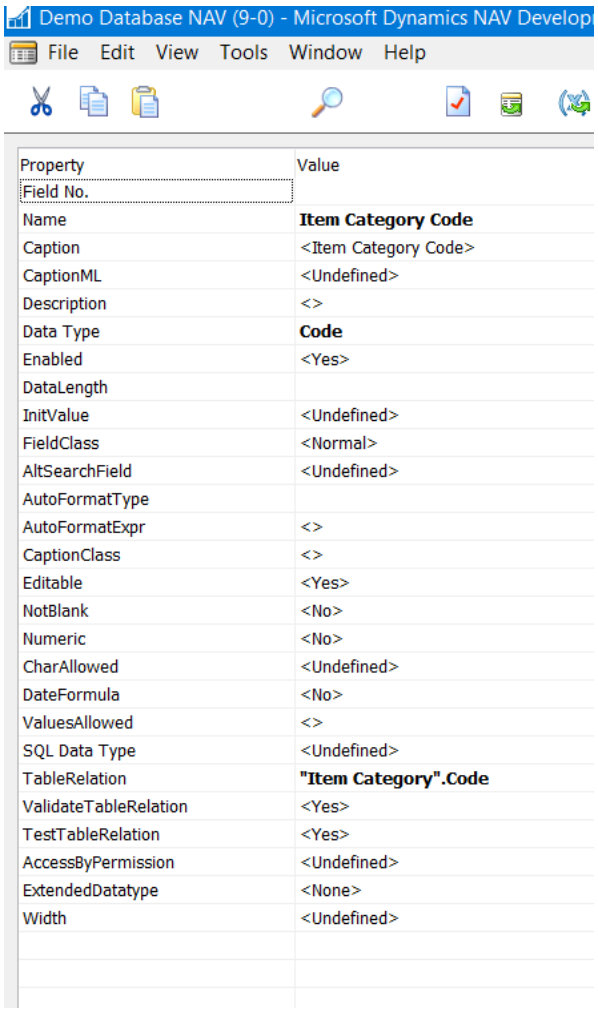
A Linking Page-en a Parameters konténerben, ami egy FactboxArea, létre kell hozni egy új sort, mely egy Part típusú sor lesz. Ennek a Subtype-ját (altípusát) itt is Page-re kell állítani. A Part inicializálása után a Property menüben be kell állítani, hogy ennek a mezőnek mi legyen a PartType-ja. Most ez Page lesz, mert egy másik Page-et használunk fel. A PagePartID property értékéhez kell megadnunk melyik Subformot használja fel, ez itt az „ItemCat. Param. Line”. Továbbá a kapcsolatot is meg kell adnunk még a Properties menüben: Item Category Code=FIELD(Code). Ezzel a beállítással az éppen vizsgált Item Category paramétereit tudjuk megjeleníteni.
Az elkészült Objektumok a következőképp néznek ki:



15. ábra: Kapcsoló tábla (Source Táble) létrehozása

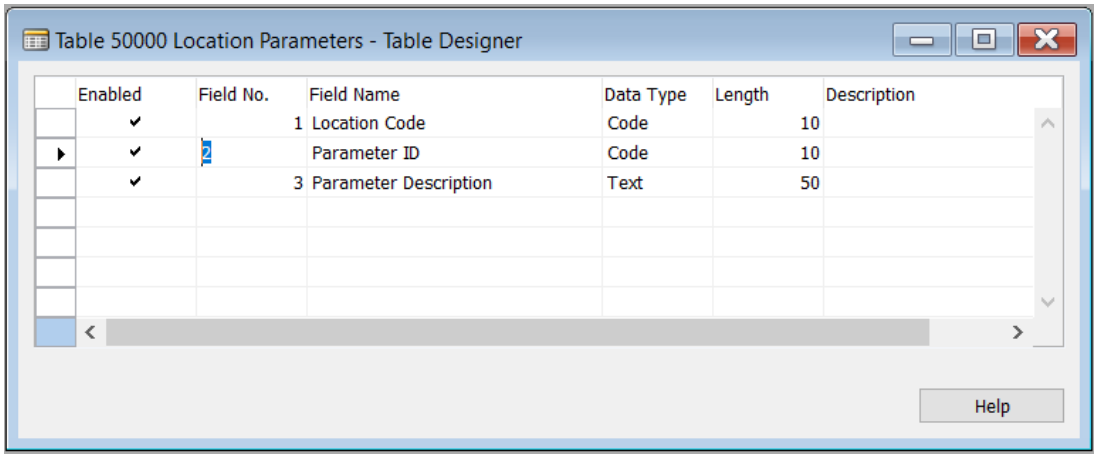


16. ábra: Kulcsok beállítása a kapcsolótáblára

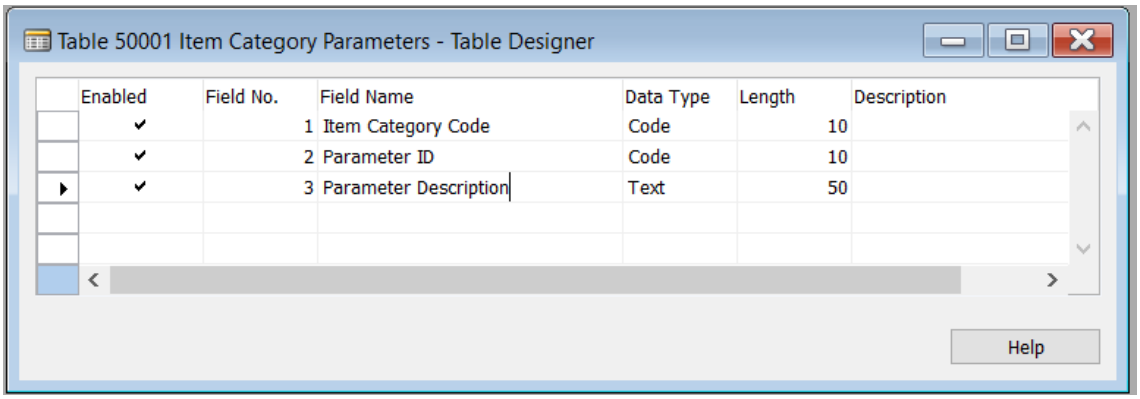


17. ábra: Item Category Code Properties, TableRealtion

A Location Code hasonlóan néz ki, ám annál a Location.Code van beállítva a TableRelation Property-nél.



18. ábra: Location Parameters Table



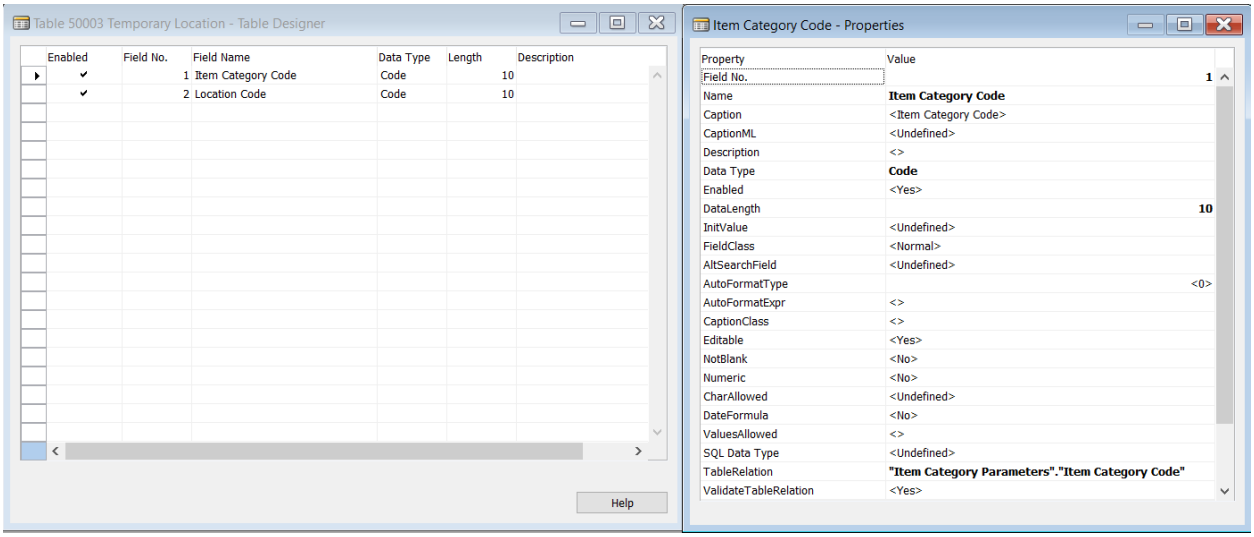
19. ábra: Item Category Parameters Table

Az előbb bemutatott két tábla kódjába, a Parameter ID – OnValidate() triggereibe írt kód biztosítja, hogy a Parameter Description-ök automatikusan töltődnek.

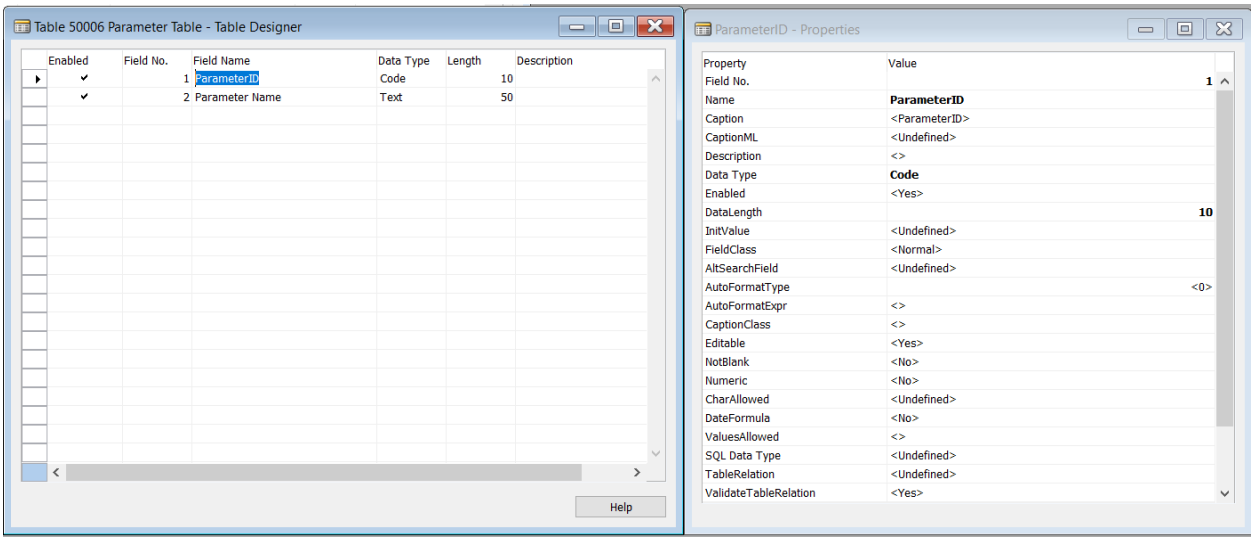
```
Parameter ID - OnValidate()
IF ParameterTable.GET("Parameter ID") THEN
  "Parameter Description" := ParameterTable."Parameter Name"
ELSE
  "Parameter Description" := '';
```

20. ábra: Parameter ID – OnValidate()

Ehhez egy lokális változót kellett felvennem mindkét triggerben. Ebben a lokális változóban a Parameter Table-t adtam át Record típusú változóként.

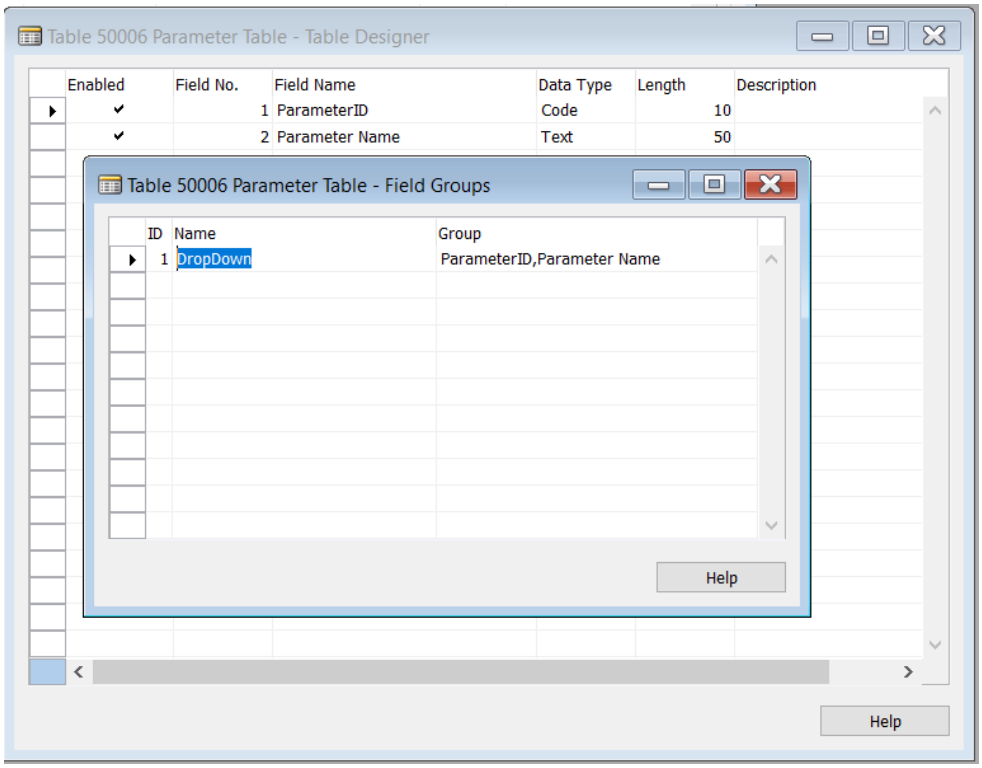


21. ábra: Temporary Location

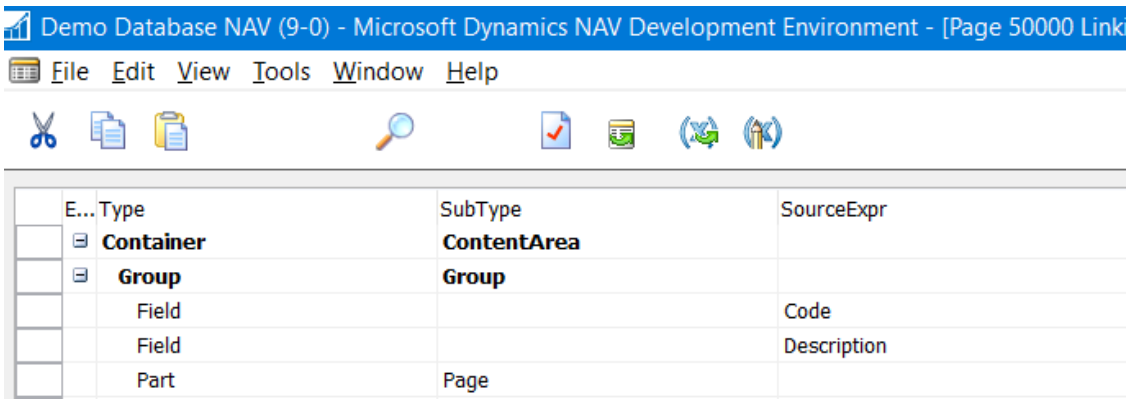


22. ábra: Parameter Table

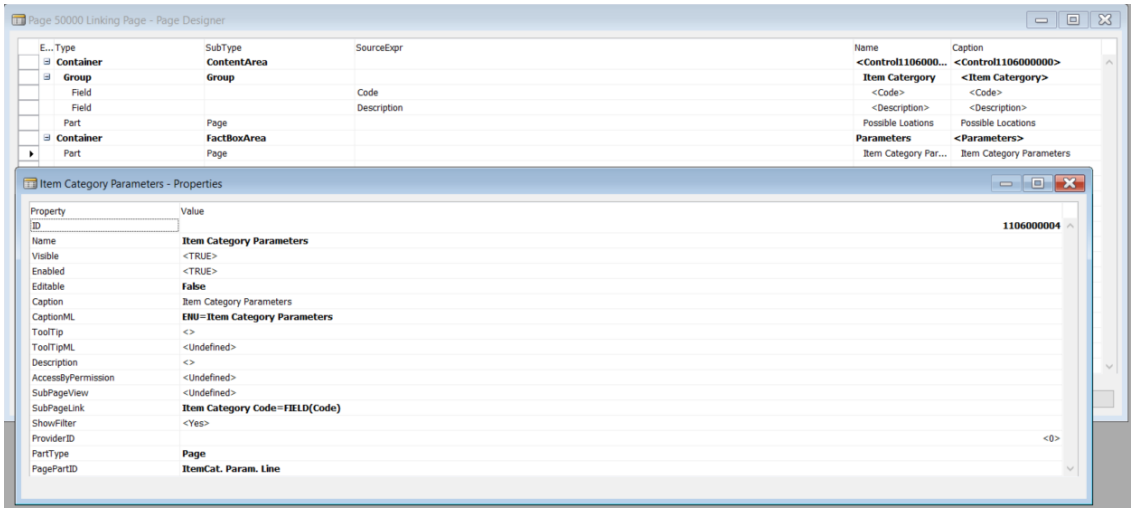
Mivel erre a táblára több másik tábla is hivatkozik, ezért célszerű beállítani egy DropDown-t. Ezt a tábla View – FieldGroup menüpontjában tudjuk megtenni. Így mikor olyan helyen szeretnénk értéket beírni, ahol a TableRealtion erre a táblára hivatkozik, láthatjuk az ID-hoz tartozó Text értékeket.



23. ábra: Parameter Table DropDown

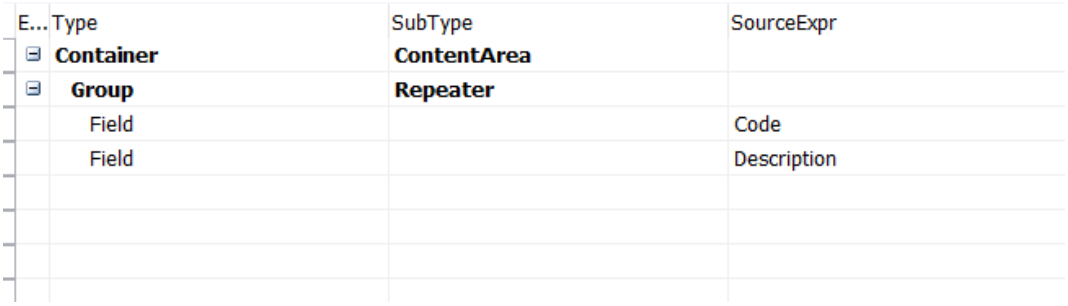


24. ábra: Adatok megjelenítése a LinkingPage-en



25. ábra: Item Category Parameters Properties

A LinkingPageList hasonlít magára a LinkingPage-re. Forrástáblája az Item Category, a két megjelenített mező a Code és a Description. A LinkingPageList-et futtatva megjelennek a cikkcsoportok és a hozzá tartotó leírások. Majd a cikkcsoportot kiválasztva, View-t megnyomva nyílik meg a Linking Page. Ehhez össze kell kapcsolni a két Page-et, a LinkingPageList Properties menüjében a CardPage ID-nál kell beállítani a Linking Page-et.



26. ábra: LinkingPageList

Property	Value
ID	
Name	**LinkingPageList**
Caption	<LinkingPageList>
CaptionML	<Undefined>
Editable	**No**
Description	<>
Permissions	<Undefined>
PageType	**List**
InstructionalTextML	<Undefined>
CardPageID	**Linking Page**
DataCaptionExpr	<Undefined>
RefreshOnActivate	<No>
PromotedActionCategoriesML	<Undefined>
SourceTable	**Item Category**
SourceTableView	<Undefined>
InsertAllowed	<Yes>
ModifyAllowed	<Yes>

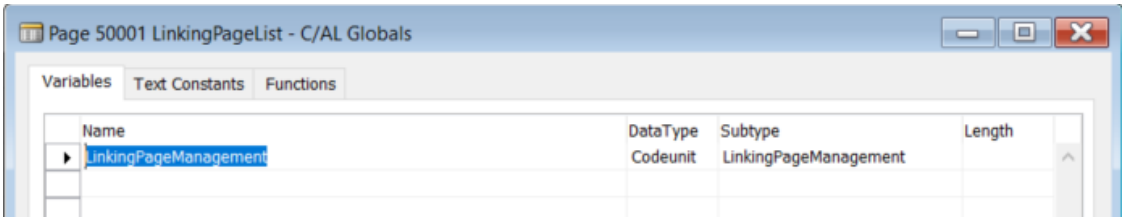
27. ábra: LinkingPageList Properties

A Linking Page gyári funkciója a Next és Previous gombok, ezek segítségével tudunk navigálni az oldalon. Ezt kihasználva tud majd a felhasználó léptetni a cikkcsoportok között.

Minden egyes az oldalon történő léptetéskor az aktuális cikkcsoportra szűrjük a táblát, így minden cikkcsoporthoz a megfelelő raktárakat jeleníthetjük meg.

Létrehoztam egy CodeUnit-ot, amiben a szükséges függvényeket tárolom. Két függvény szükséges a megvalósításhoz. Egy az összehasonlításhoz és egy a letároláshoz. Ezek név szerint a Compare és az LinkItemCatToLoc függvények. A Page Action menüben létrehoztam egy Action-t, ami majd a letároláshoz szükséges gomb lesz.

A LinkingPageList-et design módban megnyitva a Code menüben létrehoztam egy globális változót, ez CodeUnit típusú, Subtype-ja a LinkingPageManagement CodeUnit. Az OnOpenPage() triggerben példányosítva hívom meg.



28. ábra: LinkingPageList Globals

```
4
5   OnOpenPage()
6     LinkingPageManagement.Compare;
7
8   OnClosePage()
9
10  OnFindRecord(Which : Text) : Boolean
```

29. ábra: LinkingPageList Code menü – OnOpenPage() trigger

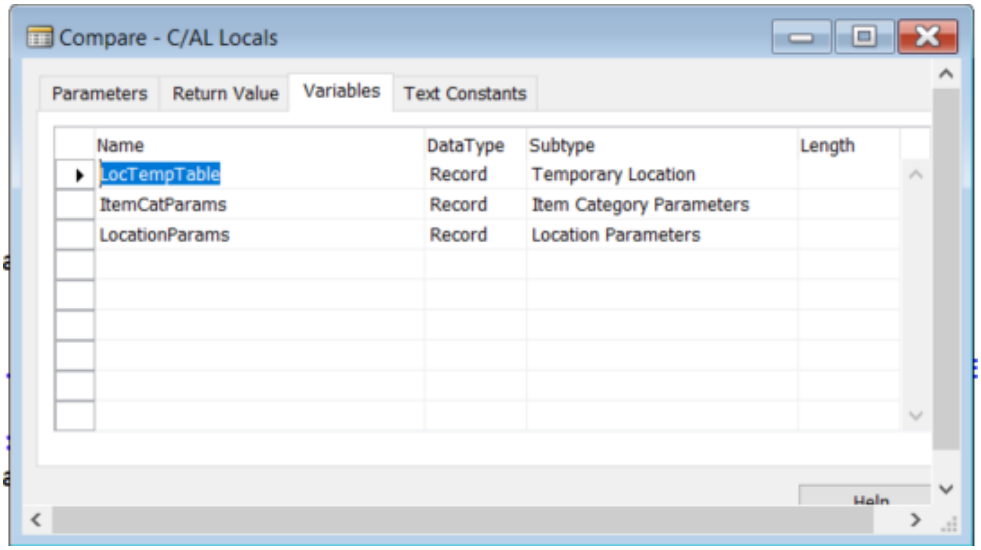
Így tehát az oldal megnyitáskor lefut az összehasonlítás az összes rekordra, és beleíródik a Temporary Location táblába, ha korábban még nem volt benne.

Fontos megjegyezni, hogy ezt azért így használjuk, mert a Header Page-eket mindig a Line típusú Page-ekből töltjük fel. És itt pontosan ez a helyzet áll fenn.

```
5  Compare()
6  //ItemCategory Parameters szűrése
7  ItemCatParams.RESET;
8
9  //Iterálás
10 IF ItemCatParams.FINDSET THEN
11   REPEAT
12     LocationParams.RESET;
13     LocationParams.SETRANGE(Parameters, ItemCatParams.Parameters);
14     IF LocationParams.FINDSET THEN BEGIN
15       REPEAT
16         IF NOT LocTempTable.GET(ItemCatParams."Item Category Code", LocationParams."Location Code") THEN BEGIN
17             LocTempTable.INIT;
18             LocTempTable."Item Category Code":=ItemCatParams."Item Category Code";
19             LocTempTable."Location Code":=LocationParams."Location Code";
20             LocTempTable.INSERT(TRUE);
21         END
22       UNTIL LocationParams.NEXT=0;
23     END
24   UNTIL ItemCatParams.NEXT=0;
25
```

30. ábra: Compare Function

A Compare Function-ben történik a paraméterek összehasonlítása. A függvényben létrehoztam három lokális változót. Mind a három Record típusú, azaz mindegyikben egy táblát adok át értékül.



31. ábra: Compare Locals

A függvény a következőképpen működik. A FINDSET gyári függvénnyel megkeressük az Item Category Parameters tábla rekordjainkat, a tábla kulcsa alapján.

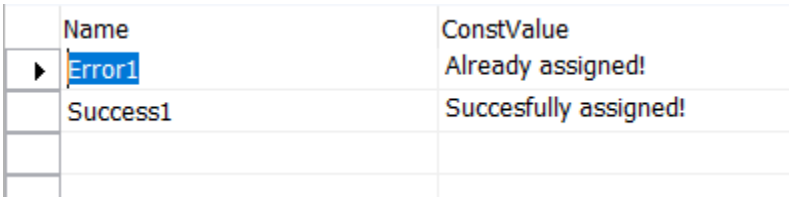
Ha ez sikerül, a Location Parameters tábláról leszedünk minden esetleg rajta maradt szűrést, rendezést (RESET), majd a tábla Paraméter oszlopát leszűrjük (SETRANGE) az Item Category Parameters tábla első megtalált rekordjának Paraméter értékére. Ezután megkeressük a táblában ezeket a rekordokat (FINDSET). Ha találunk ilyen rekordokat, megvizsgáljuk, hogy benne vannak-e már a Temporary Location táblában. Ha nincsen benne, akkor beleírjuk a táblába, ha benne van már, nem teszünk semmit, tovább lépünk. Ezt addig ismételjük míg végig nem érünk a Location Parameters rekordjain. Ezután léptetünk az Item Category Parameters tábla rekordjain. Ezt mindaddig csináljuk míg végig nem érünk ezen a táblán.

```
LinkItemCatToLoc("Item Cat. Code" : Code[10])
LocTempTable.RESET;
LocTempTable.SETRANGE("Item Category Code","Item Cat. Code");

IF LocTempTable.FINDSET THEN BEGIN
  REPEAT
    IF NOT SourceTable.GET("Item Cat. Code", LocTempTable."Location Code") THEN BEGIN
      SourceTable.INIT;
      SourceTable."Item Category Code":="Item Cat. Code";
      SourceTable."Location Code":=LocTempTable."Location Code";
      SourceTable.INSERT(TRUE);
      MESSAGE(Success1);
    END
    ELSE BEGIN
      ERROR(Error1);
    END
  UNTIL LocTempTable.NEXT=0
END
```

32. ábra: LinkItemCatToLoc Function

33. ábra: LinkItemCatToLoc Locals

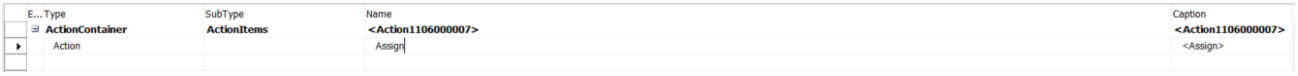


34. ábra: Text Constants

Az összerendelést végző függvénynek egy bemenő paramétere van, az oldalról kapott Item Category Code. A lefutás itt is hasonló, mint az összehasonlító függvényben. A Temporary Location tábláról leszedjük az esetlegesen rajta maradt szűréseket, rendezéseket. Ezután leszűrjük a táblát a paraméterként kapott cikkcsoportra. Így azokat a rekordokat kapjuk, amiket be kell írnunk majd a Source Table-be.

Ha találunk a szűrésnek megfelelő rekordokat, megvizsgáljuk, hogy a rekord, amit aktuálisan be akarunk szúrni, benne van-e már a táblában. Ha nincs benne, akkor beillesztjük és egy üzenettel jelezzük a felhasználónak, hogy sikeresen hozzárendelte a cikkcsoporthoz a raktárat. Ha már benne van, jelezzük neki, hogy ez az összerendelés már megtörtént. Ezt addig ismételjük amíg a leszűrt táblán végig nem érünk.

Ez a függvény a bemenő paramétert az oldalról kapja. Létrehoztam egy PageAction-t a Linking Page oldalon.



35. ábra: Assign PageAction létrehozása

A Properties-ben több dolgot is be tudtam állítani ehhez kapcsolódóan. Többek között tudtam képet beállítani a gombhoz, emellett a Promoted opció YES-re történő állításával a Ribbon-on elöl jelenik meg. Megadtam továbbá, hogy a Process kategóriába kerüljön, és nagy méretben jelenjen meg.

Property	Value
ID	
Name	**Assign**
Visible	<TRUE>
Enabled	<TRUE>
RunPageMode	**Edit**
Caption	<Assign>
CaptionML	<Undefined>
ToolTip	<>
ToolTipML	<Undefined>
Description	<>
AccessByPermission	<Undefined>
Image	**Apply**
Promoted	**Yes**
PromotedCategory	**Process**
PromotedIsBig	**Yes**
Scope	<Page>
Ellipsis	<No>
ShortCutKey	<>
RunObject	<Undefined>
RunPageView	<Undefined>
RunPageLink	<Undefined>
RunPageOnRec	**No**
InFooterBar	<No>

36. ábra: Assign Properties

Az Assign Code menüjét megnyitva találom az Action-höz tartozó triggereket, ami jelenleg csak a Documentation() és az OnAction() triggereket jelenti. Létrehoztam egy globális változót LinkingPageManagement néven. Ez egy CodeUnit típusú változó, Subtype-ja a LinkingPageManagement CodeUnit. Ezzel itt is elérem a CodeUnit-ban megírt függvényeket.

Name	DataType	Subtype	Leng
LinkingPageManagement	Codeun	LinkingPageManagement	

37. ábra: Assign – C/AL Code, Globals

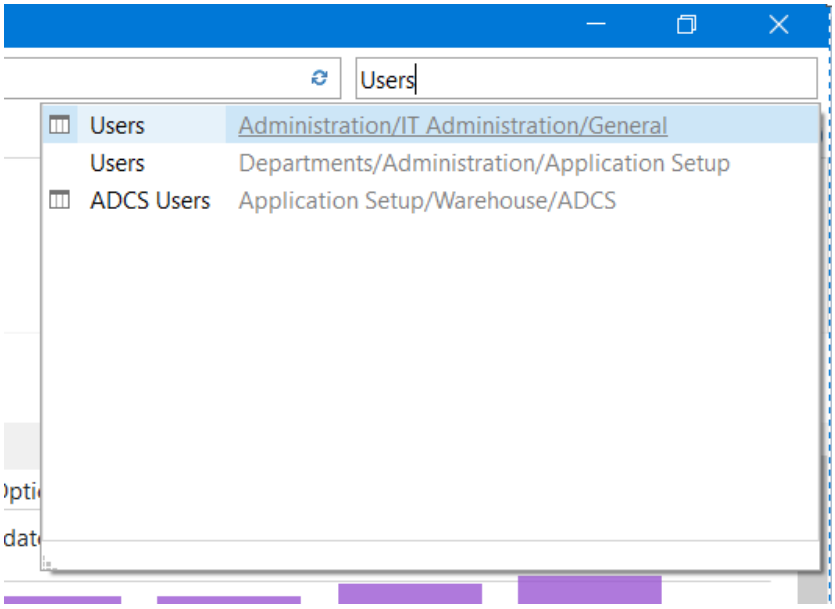
Az OnAction() triggerben példányosítva ezt a CodeUnit-ot tudom meghívni a LinkItemCatToLoc függvényt.

```
1  Documentation()
2
3  Assign - OnAction()
4  LinkingPageManagement.LinkItemCatToLoc(Rec.Code);
5
```

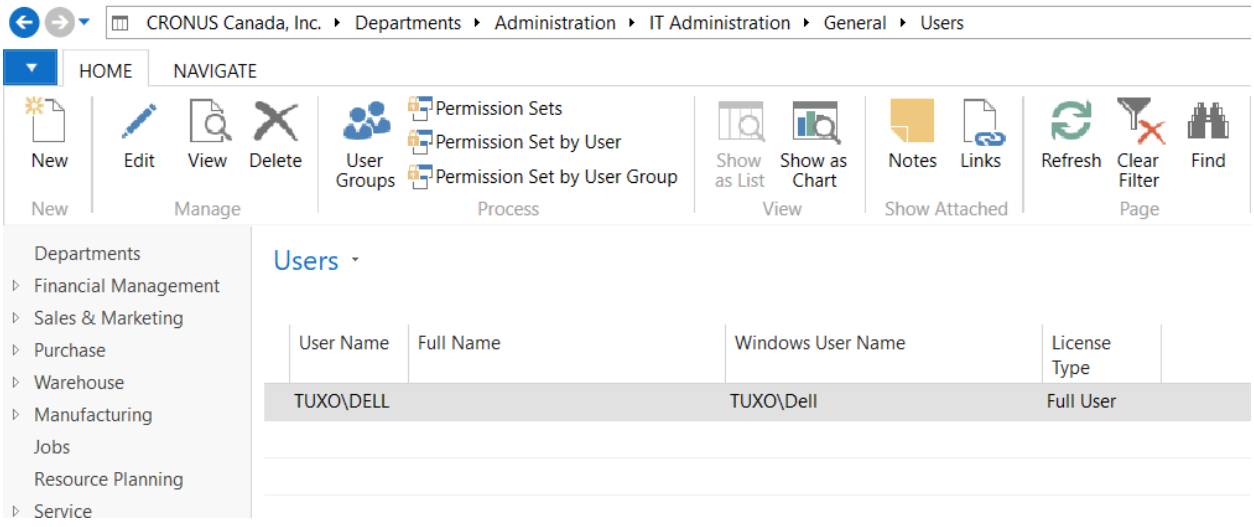
38. ábra: Assign – OnAction() triggerben LinkItemCatToLoc függvény hívása

4.2. Jogosultság kezelés megvalósítása

A NAV gyárilag nyújt felhasználó jogosultság halmazokat. A jogosultsági halmaz határozza meg, ki férhet hozzá az adott objektumhoz az adatbázisban. Minden felhasználót egy vagy több objektumhoz kell hozzárendelnünk, hogy hozzáférése legyen a NAV-hoz. Egy felhasználó jogosultságainak definiálásához a kliensünkre lesz szükségünk.
A következő lépésekkel tudjuk beállítani a jogosultságokat.

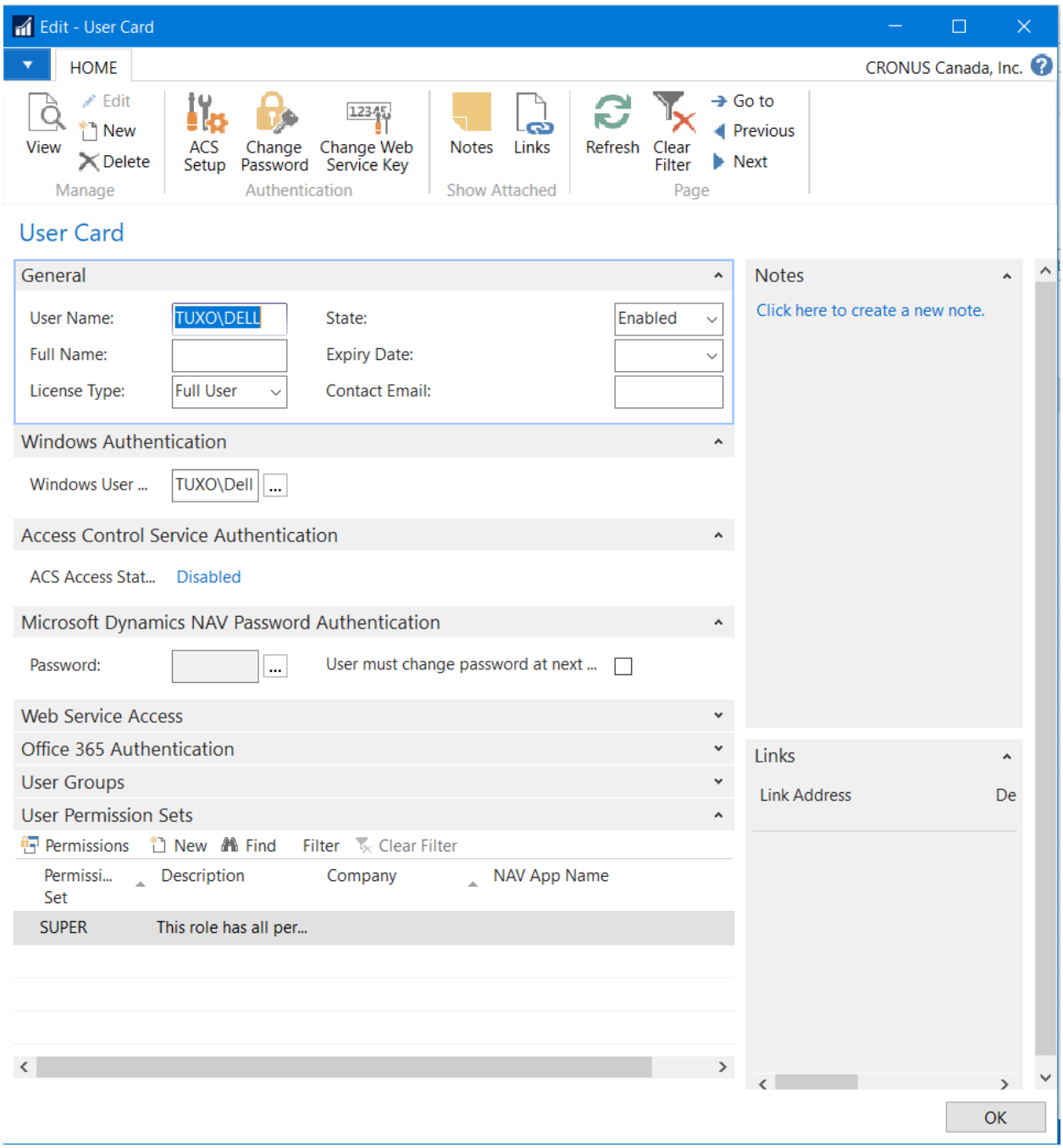


39. ábra: Users megkeresése a NAV keresőjében



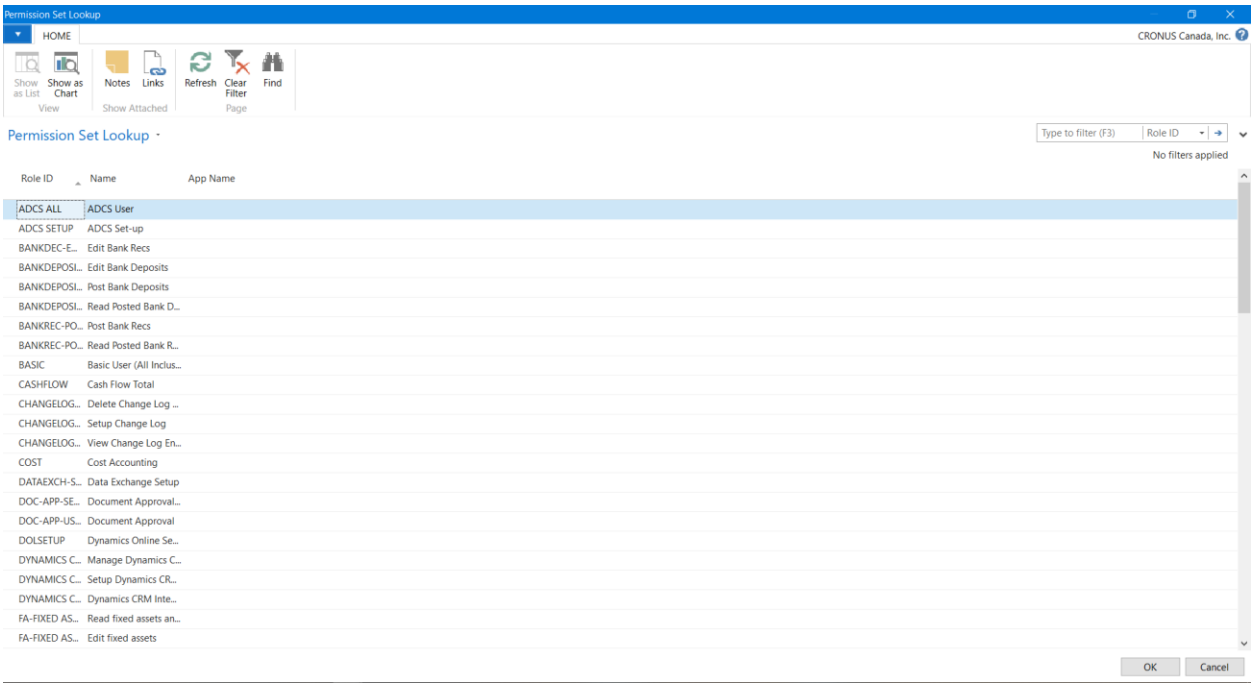
40. ábra: Felvett User-ek

Ezek után az Edit gombra kattintunk.



41. ábra: Edit – User Card

Itt beállíthatjuk, mihez férjen hozzá a felhasználónk. A saját felhasználómat a fejlesztési szakasz legelején, Témalabor tárgyamkor felvettem Full Userként. Így rendszergazdai jogosultsággal rendelkezem a programban. Természetesen ezen is könnyen változtathatunk. A User Permission Sets fül alatt, a Permission Set lehetőségnél a SUPER-en tudunk változtatni.

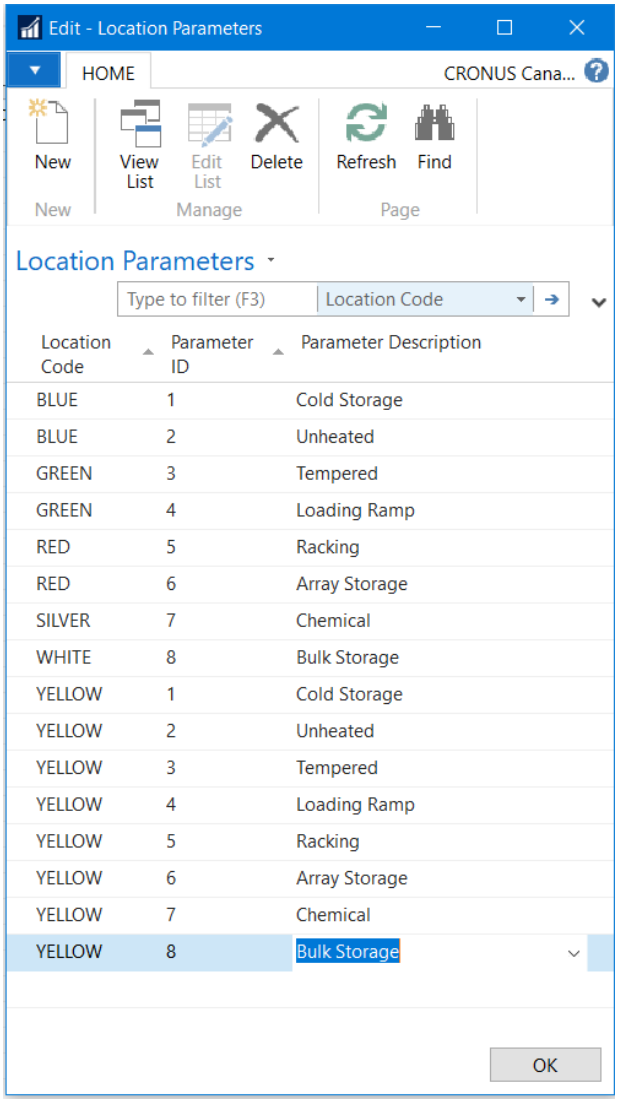


42. ábra: Permission Set Lookup

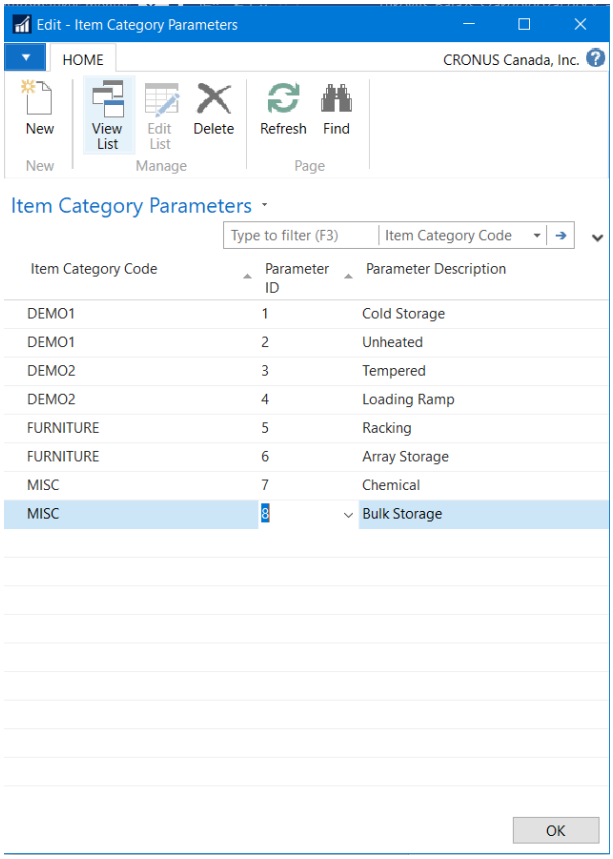
Ez a beállítás a fejlesztésem számára teljesen megfelelő. Ezt az üzleti életben természetesen az ügyfél igényei szerint lehet majd változtatni, módosítani.

5. Eredmények bemutatása

Első lépésként létrehoztam két további cikkcsoportot (DEMO1 és DEMO2) és feltöltöttem a paramétereket tartalmazó táblákat adatokkal a teszteléshez.

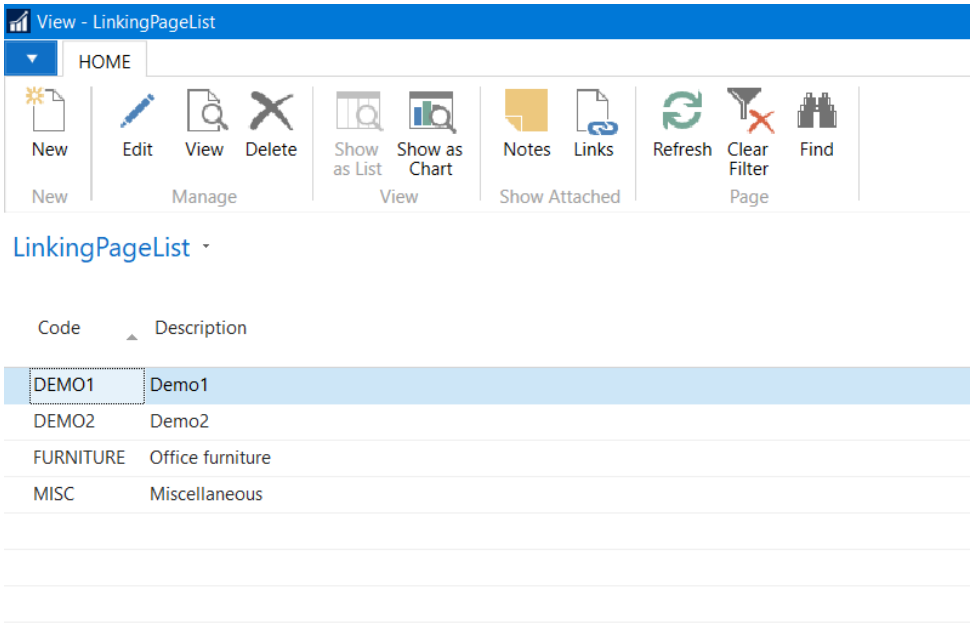


43. ábra: Location Parameters tábla feltöltve



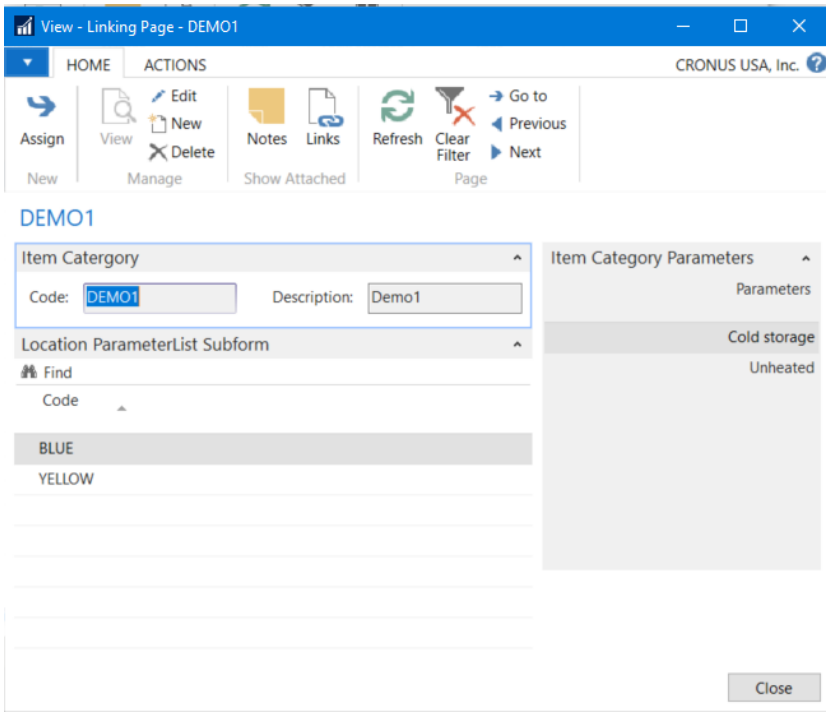
44. ábra: Item Category Parameters tábla feltöltve

Futtattam a LinkingPageList-et, majd rákattintottam egy Cikkcsoportra.

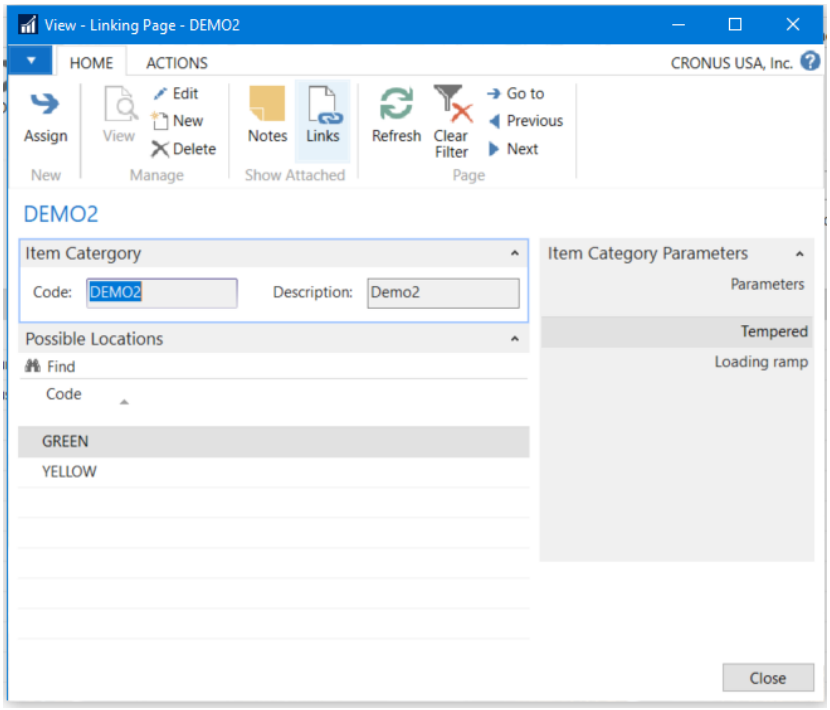


45. ábra: LinkingPageList

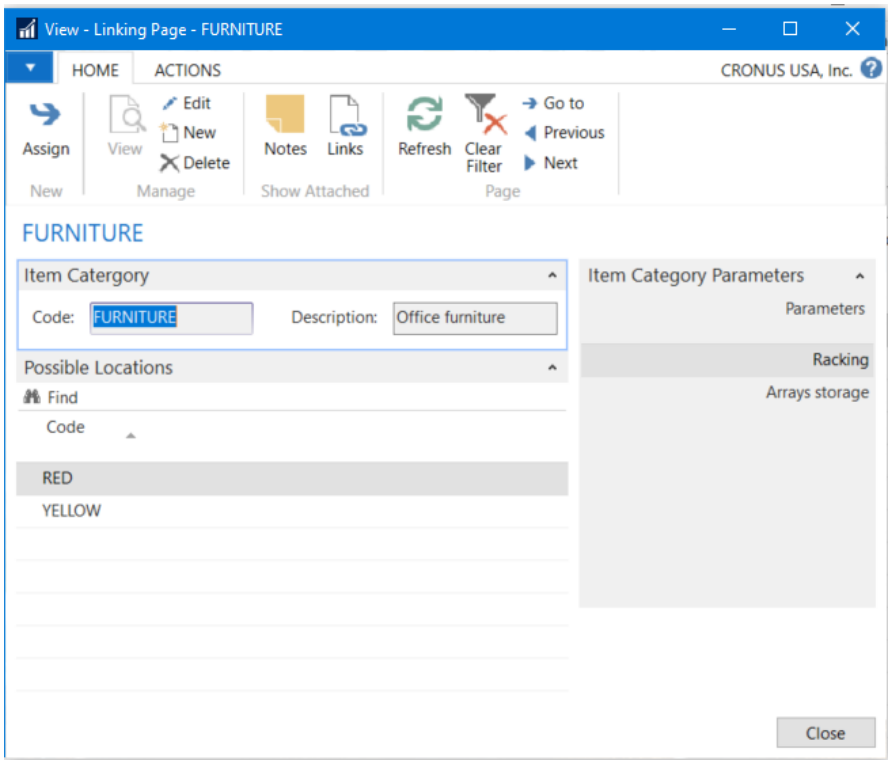
A következő lépésben a View menüpontot megnyitva megfut a Linking Page. A Next és Previous gombokkal navigálva ellenőriztem a kód helyes lefutását és a helyes kiíratást.



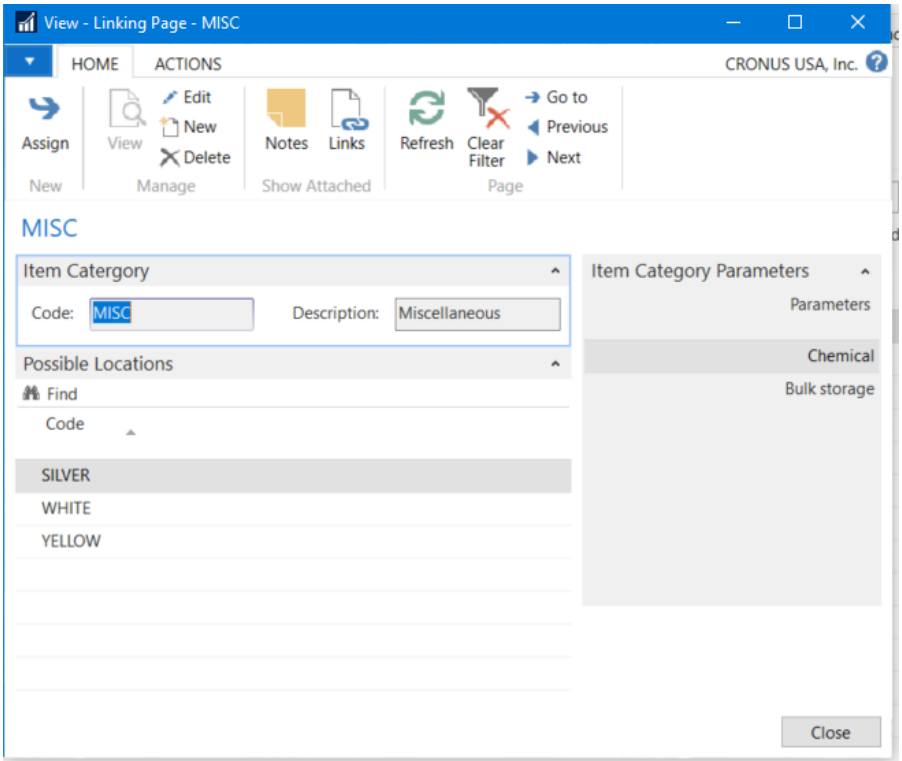
46. ábra: DEMO1 cikkcsoport, paraméterei és a szűrt Location a Linking Page-en



47. ábra: DEMO2 cikkcsoport, paraméterei és a szűrt Location a Linking Page-en

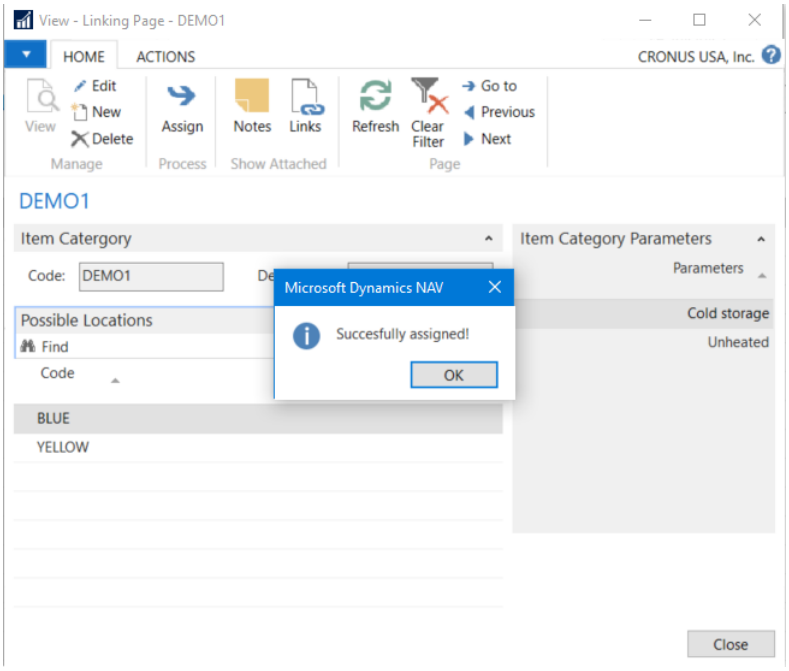


48. ábra: FURNITURE cikkcsoport, paraméterei és a szűrt Location a Linking Page-en



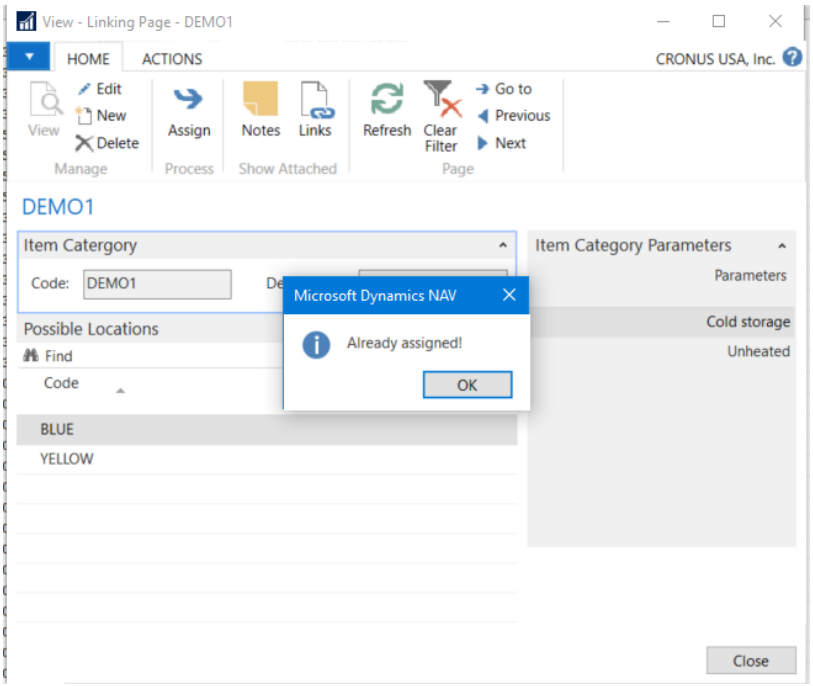
49. ábra: MISC cikkcsoport, paraméterei és a szűrt Location a Linking Page-en

Meggyőződtem róla, hogy hibátlanul megtörtént a szűrés, ezért létrehozhatom a kapcsolatot a cikkcsoport és a raktárak között.

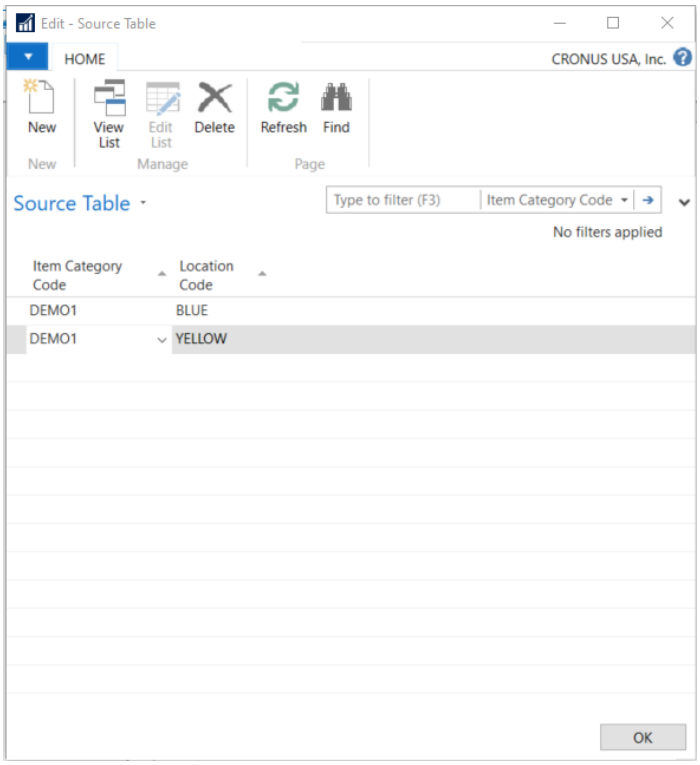


50. ábra: DEMO1 cikkcsoporthoz Location-ök rendelése

Mivel a kód figyeli, ezért nem tudunk egy kapcsolatot kétszer létrehozni.



51. ábra: DEMO1 cikkcsoportnál „kétszeres összerendelés”



52. ábra: Kapcsolatok bekerültek a Source Table-be

6. Összefoglalás

Önálló munkáim során rengeteget foglalkoztam a raktározással és annak folyamataival. Rengeteget olvastam és tájékozódtam ebben a témában. Utánajártam miképp működik ez a NAV-ban. Volt lehetőségem olyannal is beszélgetni a témáról, aki személyesen is használta ezt a rendszert. Miután megismertem a témám alapját képező elméletet, nekiláttam a szoftver megismerésének. A NAV egy nagyon jó program, ezt a piacon elfoglalt pozíciója is jól tükrözi. Fejlesztőként is könnyen tanulható és átlátható a logikája.

Tökéletesnek bizonyult a Témalabor témájának választása, ugyanis erre az önálló munkára építettem fel Önálló Labor dolgozatomat, majd pedig jelen szakdolgozatomat. A három félév alatt beleláthattam a NAV világába, megismertem a folyamatait, fejlesztésének mikéntjét.

Megismerkedtem a C/AL nyelvvel, annak nyelvi elemeivel, sajátosságaival, objektumaival.

A sikeres Témalabor beszámoló után kapott állásajánlat óriási lendületet adott, amire tudtam építeni a későbbiekben. Igaz, órai keretek között (VIR, VIR2, ERP rendszerek ipari alkalmazása és fejlesztése) is nagyon jó alapokat kaptam, ám a Cosmonál eltöltött másfél hónap is rengeteget segített elkészíteni a szakdolgozatomat. Sok tapasztalatot szereztem, jobban megismertem a környezetet és a nyelvet. Elsajátíthattam egy olyan rendszerszemléletet, melyet önállóan sosem sikerült volna megismernem.

A docs.microsoft.com oldal is óriási segítségemre volt, amiben egyre jobban és egyszerűbben igazodom el.

A feature elkészítéséhez adatbázis módosítást kellett végrehajtanom. Táblákat, Page-eket, Codeunitot kellett létrehoznom. Utána kellett járnom a NAV jogosultság kezelésének és annak működésének. Az általam létrehozott táblákat fel kellett töltenem teszt adatokkal, hogy tudjam ellenőrizni a helyes lefutást.

Természetesen, mint minden kiegészítő, ez is tovább „faragható”. Önálló Labor dolgozatom után Demoként utaltam munkámra. Most szakdolgozat keretei között ez már egy kész alkalmazás, amelyet bármely ügyfél rendszerébe képesek vagyunk implementálni és kérésének megfelelően tulajdonképpen bármivel bővíteni.

7. Ábrajegyzék

8. Irodalomjegyzék

[1] Logisztika jegyzet, Debreceni Egyetem http://www.agr.unideb.hu/ebook/logisztika/index.html (Utolsó megtekintés: 2020. szeptember 26.)
[2] Logisztika I-II, egyetemi jegyzet, Széchenyi Egyetem, Győr, 2006
[3] Hans van der Hoeven: ERP and Business Processes, Llumina Press, 2009.
[4] Digitális Tankönyvtár, Logisztikai alapismeretek https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0085_logisztikai_alapismeretek/ch02.html#id500766 (Utolsó megtekintés: 2020. szeptember 26.)
[5] Korszerű raktározási rendszerek, Buczkó Balázs, szakdolgozat, Miskolci Egyetem, Anyagmozgatási és Logisztikai tanszék, 2013
[6] HVG.hu, Embernek be sem kell mennie az új Hell automata gigaraktárba c. cikk, https://hvg.hu/kkv/20190212_hell_magasraktar?fbclid=IwAR0a3q1WTAlvV9zquRQtvAqV47h_3lbK8pqQt4HSSZp87sjvuPlUHdyMWj0 (Utolsó megtekintés: 2020. szeptember 29.)
[7] Vállalat irányítási rendszerek tárgy, Full time – Sales laborjegyzet, Dr. Bencsik Gergely, Soproni Egyetem
[8] Microsoft Dynamics NAV 2016 dokumentáció: https://docs.microsoft.com/en-us/dynamics-nav-app/ (Utolsó megtekintés: 2020. november 20.)
[9] ERP rendszerek ipari alkalmazása és fejlesztése tárgy órai anyagai