# Szigetrendszer koncepció bemutató rendszer

Egy egyszerű lépésekből álló logisztikai mini-folyamaton akarjuk bemutatni a szigetrendszerek koncepciót.

Minden egyes lépés végrehajtása egy munkahelyen történik, amelyeket egy-egy szigetmodul kezel.

A visszakövetés szigetmodul a munkahelyi sziget modulok „hátterében” működik, és mindegyik munkahely számára adatokat szolgáltat és a munkahelyi tevékenységek alapján adatokat regisztrál. A munkahelyi sziget modulok egymással is beszélgetnek.

A sziget modulok közötti kommunikáció érdekében a sziget modulok interfészt biztosítanak, amelyen keresztül a modul logisztikai tranzakciói hajthatók végre, és amelyen keresztül adatok kérdezhetők le. A modulokhoz hozzáférni máshogy, mint ezen interfészen keresztül, nem lehet.

## Áttekintés

|  |  |  |  |
| --- | --- | --- | --- |
|  | Sziget modul neve  Munkahely / munkalépés | A bemutató rendszer által elvégzett tevékenység | Megjegyzések |
| 1 | Log4ProIS- TAKEIN  Szállítási egység bevételezése a raktárba | Az anyagátvételi területen félautomata technikával (2D kamerával) leolvassák a beérkező szállítási egységen levő beszállítói adatokat, amely alapján egy belső azonosítókat és a mennyiséget tartalmazó, vonalkóddal és UHF RFID bélyeggel ellátott etikett kerül kinyomtatásra. A munkahelyi tevékenység eredményeképpen létrejön és bevételezésre kerül a szállítási egység a rendszerben. A szállítási egység azonosítója a bevételezéskor generált azonosító és az etikettre kinyomtatott vonalkódba és az etikettben levő RFID bélyegbe is beírásra kerül. | A bevételező munkahely monitor képernyőjén megjelennek a státusz adatok. Ezt a munkahelyet a ReelCheck szoftverrel kezeljük, amely áthív a visszakövetés modulba (ehhez a ReelCheck kismértékű továbbfejlesztése, a RedisEventHub interfész kezelésének beépítése, szükséges, amely nagyon passzol a ReelCheck jelenlegi funkcionalitsához, és jól kiegészíti azt.). |
| 2 | Log4ProIS-RECEIVING  Szállítási egység betárolása a raktárba szállítószalagon | A bevételezett szállítási egységet egy szállító­sza­lagon bemozgatják a raktárba. A szállítószalag UHF RFID leolvasója regisztrálja az átmozgó egységet, a modul kioszt a részére egy raktári-tárolóhely kódot, és a szállítási egységet betárolt státuszba helyezi. A betároló munkahely monitor képernyőjén a művelet elvégzése után megjelenik a szállítási egység azonosítója, a cikkszám és a mennyiség, valamint a kiosztott tárolóhely kód. | A Füzi Krisztián féle RFID szoftver az alapja a munkahely kezelő szoftverének. Ehhez készíteni kell egy kiegészítő komponenst, amely az RFID leolvasóval beolvasott vonalkód alapján elvégzi a szükséges műveleteket, amely áthív a visszakövetés modulba. |
| 3 | Log4ProIS-REPACKING  Szállítási egységből csomagolási egységbe való átcsomagolás a raktárban | A termelésbe csak sztenderd csomagolású egy­ségek kerülhetnek ki, ezért a kiadás előtt ezeket a csomagolási egységeket el kell készíteni. A cso­ma­golási egységek azonos adattartalmú (a cso­magolási egység azonosítója) HF RFID, UHF RFID és vonalkódos jelöléssel rendelkező, újrafel­hasz­nálható egységek. A csomagoló munkahelyen leolvasásra kerül vonalkóddal a forrás szállítási egység azonosítója, a cél szállítási egység azonosítója, valamint az áthelyezett mennyiség.  Az átcsomagoló munkahely monitor képernyőjén a művelet elvégzése után megjelenik a forrás szállítási egység azonosító, a benne levő mennyiség, a csomagolási egység azonosítója | Ennek a munkahelynek a kezeléséhez egy web-POST akciót kell készíteni a megfelelő url paraméterezéssel. Ennek az akciónak a meghívása a WebForm programmal (megfelelő xml-paraméterezéssel) történik.  Az akció áthív a visszakövetés modulba. |
| 4 | Log4ProIS-PUTOUT  Csomagolási egység kitárolása a raktárból szállítószalagon | A létrehozott csomagolási egységet egy szállítószalagon kimozgatják a raktárból. A szállítószalag UHF RFID leolvasója regisztrálja az átmozgó egységet.  A betároló munkahely monitor képernyőjén a művelet elvégzése után megjelenik a szállítási egység azonosítója, a cikkszám és a mennyiség. | A Füzi Krisztián féle RFID szoftver az alapja a munkahely kezelő szoftverének. Ehhez készíteni kell egy kiegészítő komponenst, amely az RFID leolvasóval beolvasott vonalkód alapján elvégzi a szükséges műveleteket. A betárolásban alkalmazott funkcióhoz hasonló kerül itt is kialakításra, de amíg a betároláskor egy szállítási egység azonosító, a kitároláskor egy csomagolási egység azonosító kerül vele leolvasásra. Ezeknek felismerhetően különbözniük kell egymástól (pl. prefix), és ez alapján kell eldönteni, hogy melyik műveletet (értsd: betárolás, vagy kitárolás) kerül végrehajtásra. A modul áthív a visszakövetés modulba és behív a kan-ban modulba is. |
| 5 | Log4ProIS-KANBAN  Csomagolási egység elhelyezése kanban-állványon a gyártó területen | A sztenderd csomagolási egység elhelyezése a termelési területen elhelyezkedő kan-ban polcrendszerre. A Kitárolás művelet végrehajtása után a kan-ban monitor képernyőjén, amely a kan-ban állvány térképét tartalmazza, az éppen kitárolt csomagolási egységet fogadó cella sárgára vált (fehérről, ami az üres cellát jelenti), jelezve, hogy egy érkező csomagolási egység számára lefoglalásra került. A cellában megjelenik a várt csomagolási egység azonosítója. A csomagolási egységet be kell helyezni a kan-ban állványba, a sárgával jelzett cellába. Ha a betárolás a megfelelő helyre történt, akkor a cella zöldre vált (ok), ha nem, akkor pirosra, ami téves tartalmat jelent. Ha a csomagolási egységet kiveszik az állványról, akkor a cella fehérre vált. (Ez nyilván nagyon laboratóriumi, de most elég lesz). | A kanban modul egy szerviz formát ölt, amely interfészt biztosít egy érkező csomagolási egység regisztrálását végző művelet elvégzéséhez, amely művelet a betárolás műveletben kerül meghívásra. |
| X | Log4ProIS-TRM | Nem tartozik hozzá közvetlenül logisztikai munkahely és logisztikai lépés, hanem a folyamat hátterét képviseli |  |

## Sziget - modulok közötti kommunikáció

A modulok közötti kommunikáció az un. RedisEventHub segítségével un. csatornákon keresztül történik. Más lehetőségek is vannak, ebben a demo rendszerben, egyelőre, csak azt használjuk, ahol a hívást fogadó (szerver) sziget modul nyitja meg a csatornát, amelybe kéréseket bármelyik másik sziget modul (feladó, kliens) feladhat. Ez a modell egy adatcsomag átadását és a válasz adatcsomag vételét biztosítja a feladó sziget modul számára - tehát egy klasszikus függvényhívásként kell használni, amelyben meg kell nevezni a szerver fogadó modul által megnyitott csatornát.

Lehetőség van „broadcast” típusú kommunikációra is (ezt most itt nem használjuk), amikor egy sziget modul (a hívó) egy eseményt jelez egy általa megnyitott csatornán (beledobva az esemény paramétereit), és az erre a csatornára feljelentkezett minden más sziget modul (fogadó) értesül az esemény bekövetkezéséről. Ilyenkor az eseményt feladónak nem kell válaszolnia és a hívó ezáltal nem kap semmiféle választ a hívására.

## A sziget-modulok

|  |  |
| --- | --- |
| Log4ProIS-TRM | A visszakövetés adatbázis modul (Log4ProIS-TRM – Log4Pro Island System – Traceability Modul) egy ApplicationContainer Windows Service-hez készült plugin formáját ölti, amely a fenti RedisEventHub interfészen keresztül fogadja a bejövő hívásokat.  A következő típusú hívások vannak: (1) Beavatkozások (iv-intervention): tényleges státusz változtatást végeznek, amelyek egy SQL adatbázisban kerülnek átvezetésre.  (2) Lekérdezések (qu-query): a bemenő paraméterek alapján adatokat szolgáltatnak, státuszt (SQL adatbázis) változás nem kapcsolódik hozzájuk.  (3) Monitor-adat regisztrációja (mr – monitor-data registration): most az egyszerűség kedvéért a visszakövetés modulban alakítjuk ki azokat a beavatkozásokat is, amelyek a monitor adatok átvételét és a megfelelő elhelyezését biztosítják; a monitor adatokat SQL adat táblában helyezzük el, kiolvasásuk a WebMonitor programmal történik.  Ennek a feladatnak a környezete adott, jelenleg a plugin belsejét, az egyes műveletekhez tartozó egyszerű üzleti logikát kell elkészíteni. A RedisEvenHub elérése a beavatkozást küldő oldalon, a beavatkozások fogadása a plugin-en belül, monitor adatok kiírása a Redis alapú monitor adattárolóba (ezt csak akkor, ha marad rá idő) kipróbált technikák, a ReelCheck szoftverben ezt alkalmazzuk, és Zoli jelenleg a Log4Pro-IS-ANM modulon (Andon) pontosan ezzel a technikával dolgozik, így a fejlesztési keretet (az „üres plugin-t”) ő fogja előkészíteni.  A visszakövetés modul alábbiakban összefoglalt műveleteit a sziget-modulok hívják az ott leírtak szerint:   * Log4ProIS-TRM / ivWTAKEIN: SU létrehozása és „BEVÉTELEZVE” státuszba helyezése * Log4ProIS-TRM / ivWRCV: SU betárolásának regisztrálása, „BETÁROLVA” helyezése és raktári tárhely hozzárendelése. * Log4ProIS-TRM / mrWRCV: SU betárolás munkahely monitor adatainak kiírása * Log4ProIS-TRM / ivREPCK: PU létrehozása és „LÉTREHOZVA” státuszba helyezése * Log4ProIS-TRM  / mrREPCK: SU-PU átcsomagolás munkahely monitor adatainak kiírása * Log4ProIS-TRM / ivPUTOUT: PU kitárolás regisztrálása, PU „KITÁROLVA” státuszba helyezése * Log4ProIS-TRM / ivKBIN: PU betárolása kan-ban állványra, PU „KB-ÁLLVÁNYON” státuszba helyezése. * Log4ProIS-TRM / ivKBOU: PU kitárolása kan-ban állványról, PU „TERMELÉSBEN” státuszba helyezése. |
| Log4ProIS-TAKEIN  Szállítási egység bevételezése a raktárba | * A Reelcheck szoftverrel leolvasásra kerülnek a szállítási egységen (SU-shipping unit) lévő címkén szereplő beszállítói adatok: beszállítói cikkszám (ExPN – external part number),  beszállítói szállítási egység azonosító (ExSUId – external SU id), mennyiség (SUQty). * A ReelCheck lekérdezi a kapcsolatos adatokat a Log4ProIS-TRM modultól a quWTAKEIN lekérdezéssel  quWTAKEIN bemenő paraméterei: ExPN,ExSUId, SUQty,  quWTAKEIN válasz adatok: belső cikkszám (InPN – internal part number), szállítási egység belső azonosító (InSUId – internal SU id), mennyiség (SUQty). * A Reelcheck kinyomtat egy etikettet (ami UHF RFID bélyeget is tartalmaz), amely vonalkódosan és RFID-val a szállítási egység azonosítóját (InSUId) tartalmazza. * A ReelCheck végrehajtja a betárolást az ivWTAKEIN beavatkozással, ami az adatokat eltárolja a visszakövetés adatbázisba, és ehhez a SU-hoz hozzárendeli a „BEVÉTELEZVE” státuszt. ivWTAKEIN bemenő paraméterei: InPN, InSUId , SUQty, ivWTAKEIN válasz adatok: -. |
| Log4ProIS-RECEIVING  Szállítási egység betárolása a raktárba szállítószalagon | * A szállítószalag mellé telepített RFID olvasóval leolvasásra kerül az SU RFID bélyege (InSUId), amely adatot a FÜZI-RFID () szoftverhez írt feldolgozó FÜZI-WRCV program kezel. * A FÜZI-WRCV program végrehajtja az ivWRCV beavatkozást, amely rögzíti a „BETÁROLVA” státuszt a SU-hoz, és hozzárendel egy raktári tárhely azonosítót (WLOCId),  amelyet szintén az adatbázisban letárol ehhez a szállítási egységhez. ivWRCV bemenő paraméter: ISUId,  ivWRCV válasz adatok: InPN,SUQty,WLOC, amelyeket a FÜZI-INTAKE program eltárol a „Betárolás” munkahelyhez tartozó  monitor adatrekordban az mrWRCV művelettel. * A munkahelyen elhelyezésre kerül egy kijelző (web browser), amelyen a monitor adatrekord tartalmát a WebMonitor programmal megjelenítjük. |
| Log4ProIS-REPACKING  Szállítási egységből csomagolási egységbe való átcsomagolás a raktárban | * Ez a funkció még nincs elkészítve, ezt egyedileg kell kialakítani. * Áttesszük a SU tartalmának egy részét egy másik dobozba (PU – packaging unit), amin egy HF RFID etikett (kanban polcrendszerhez fog kelleni), egy UHF RFID etikett  (szállítószalagos olvasóhoz fog kelleni) és egy vonalkód található, mindegyik ugyanazzal az adattartalommal (IPUId – internal packaging unit id). * Az átcsomagolás műveletet az ISUId, az IPUId vonalkódok leolvasása és egy mennyiségi (PUQty) adat billentyűzetes megadása képviseli. * Végrehajtásra kerül az ivREPCK beavatkozás, amely létrehozza a csomagolási egységet a bele helyezett mennyiséggel (státusza „LÉTREHOZVA”) , valamint lecsökkenti a  SU mennyiséget. ivREPCK bemenő paramétere: ISUId, IPUId, PUQTy ivREPCK válasz adatok: IPN,ISUId,IPUId,SUQty(régi),SUQty(új),PUQty, a válasz adatokat elhelyezi az „Átcsomagolás” munkahelyhez tartozó monitor adatrekordban az mrREPCK művelettel. * A beavatkozás végrehajtásához az adatbeviteli felületet a WebForm program biztosítja. Ezzel meghívásra kerül egy web-post hívás, ami az ivREPCK beavatkozás input  és output paramétereinek megfelelő adatokat vár/adatokkal tér vissza. * A munkahelyen elhelyezésre kerül egy kijelző (web browser), amelyen a monitor adatrekord tartalmát a WebMonitor programmal megjelenítjük. |
| Log4ProIS-PUTOUT  Csomagolási egység kitárolása a raktárból szállítószalagon | * A szállítószalag mellé telepített RFID olvasóval leolvasásra kerül a PU UHF RFID bélyege (InPUId), amely adatot a FÜZI-RFID () szoftverhez írt feldolgozó FÜZI-WPUTOUT program kezel. * Megjegyzés: Több szállítószalagunk nem lévén, ezt a műveletet ugyanazon hajtjuk végre, mint a Betárolás műveletet, ezért azt, hogy a FÜZI-WRCV, vagy a FÜZI-WPUTOUT programot kell  végrehajtani, a beolvasott RFID bélyegben levő információ alapján kell felismerni (ez legyen az ISUId és az IPUId azonosítók eltérő prefixe). * A FÜZI-WPUTOUT program végrehajtja az ivWPUTOUT beavatkozást, amely rögzíti a „KITÁROLVA” státuszt a PU-hoz.  ivWPUTOUT bemenő paraméter: IPUId,  ivWPUTOUT válasz adatok: InPN,PUQty, amelyeket a FÜZI-PUTOUT program eltárol a „Kitárolás” munkahelyhez tartozó  monitor adatrekordban a mrWPUTOUT művelettel. * A munkahelyen elhelyezésre kerül egy kijelző (web browser), amelyen a monitor adatrekord tartalmát a WebMonitor programmal megjelenítjük. |
| Log4ProIS-KANBAN  Csomagolási egység elhelyezése kanban-állványon a gyártó területen | * Windows service. Kezeli a kan-ban térképet, a cellákhoz hozzárendelve a státusz és tartalom adatokat. * Egy cella lehet (1) inaktív, (2) szabad, (3) foglalt/üres, (4) foglalt/teli/OK és (5) foglalt/teli/NOK * Külső interfészt biztosít a PUREG művelet elérésére, amellyel egy érkező PU-t regisztrál (egy foglalt/üres cellát keres (KBLOCId) és átállítja foglalt/üres státuszúvá, a cellához rendelve a várt PU azonosítóját). Ezt az interfészt szintén a RedisEventHub segítségével kell megvalósítani, és ha belefér az időbe, akkor ezt tegyük is meg már most. * Amikor egy PU-t behelyeznek egy cellába, leolvasásra kerül annak HF RFID azonosítója. Ha a megfelelő cellába helyezték, akkor a cella státuszát foglalt/teli/OK-ra állítja, ha egy rossz cellába helyezték, akkor pedig foglalt/teli/NOK státuszúra (bejegyzi a ténylegesen betárolt PU adatait is a cellába). * Amikor kivesznek egy PU-t egy cellából, akkor a cella státuszát szabad-ra állítja és törli a tartalom információkat. * A betároláskor meghívja az ivKBIN(IPUId, KBLOCId) műveletet, és a PU státuszát „KB\_ÁLLVÁNYON”-ra állítja. * A kitároláskor meghívja az ivKBOUT(IPUId, KBLOCId) műveletet, és a PU státuszát „TERMELÉSBEN”-re állítja. * A betároláskor / kitároláskor meghívja a mrKB műveletet a monitor adatok regisztrálásához (tulképp a cella foglaltsági térkép kerül monitor adatként is kiírásra). |

A fentiek olyan irányt képviselnek, amelyek alkalmazásra kerülnek az éles rendszerekben is, ezért a megvalósításban semmi felesleges munka sincs, és remélem a megvalósítás (sok) ideje sem indokolja hogy kényszer megoldásokat találjunk.

## Adatbázis terv-előzetes

|  |  |
| --- | --- |
| SUPPLIERS  A beszállítók adatait tartalmazó törzsadat tábla | * Id: a beszállító azonosítója * Description: megnevezés |
| PARTS  A cikktörzs adatokat tartalmazó tábla | * InPN: Belső cikkszám * ExPN: Külső, vagy beszállítói cikkszám * SupplierId: a beszállító azonosítója (egy SUPPLIERS rekordra hivatkozik) * Description: megnevezés |
| SHIPPINGUNITS  A szállítási egységek pillanatnyi és korábbi adatait tartalmazó tábla | * Id: a rekord azonosítója * Active:  =true:azt jelzi, hogy az aktuális állapotot ez a rekord tartalmazza-e (egy ilyen lehet), =false:egy korábbi állapotot tartalmaz * ISUId: belső szállítási egység azonosító * ESUId: külső szállítási egység azonosító (a beszállító címkén szerepelhet) * InPN: a töltő cikk cikkszáma (egy PARTS rekordra hivatkozik) * SUQty: a tartalmazott mennyiség * Status: az SU státusza (a leírásban szereplő értékekkel) * PrevTRXId: annak a módosító tranzakciónak az azonosítója, amely ezt a rekordot Active=true státusz értékkel létrehozta (egy TRX rekord azonosítója). * NextTRXId: annak a módosító tranzakciónak az azonosítója, amely ezt a rekordtartalmat felhasználva a következő állapotot előállította  és ebben a rekordban az Active=false státusz beállítást elvégezte (egy TRX rekord azonosítója). |
| PACKAGINGUNITS  A csomagolási egységek pillanatnyi és korábbi adatait tartalmazó tábla | * Id: a rekord azonosítója * Active: =true:azt jelzi, hogy az aktuális állapotot ez a rekord tartalmazza-e (egy ilyen lehet), =false:egy korábbi állapotot tartalmaz * IPUId: belső szállítási egység azonosító * InPN: a töltő cikk cikkszáma * PUQty: a tartalmazott mennyiség * SHIPPINGUNITSId: annak a szállítási egységnek az azonosítója (valójában a leíró rekordjának az Id-ja), amiből ezt a PU-t feltöltötték (egy SHIPPINGUNITS rekordra hivatkozik) * Status: az SU státusza (a leírásban szereplő értékekkel) * PrevTRXId: annak a módosító tranzakciónak az azonosítója, amely ezt a rekordot Active=true státusz értékkel létrehozta (egy TRX rekord azonosítója). * NextTRXId: annak a módosító tranzakciónak az azonosítója, amely ezt a rekordtartalmat felhasználva a következő állapotot előállította és ebben a rekordban az Active=false státusz beállítást elvégezte (egy TRX rekord azonosítója). |
| TRANSACTIONS  A végrehajtott műveleteket tartalmazó tábla | * Id: rekord azonosító; minden művelet végrehajtásakor létrejön egy új TRX rekord. * TS: a művelet végrehajtásának ideje * Type: a művelet típusa (a leírásban szereplő ivXXX értékek egyike) |
| SUTRX  A SHIPPINGUNITS tábla változásai | * Id: a rekord azonosítója * TRANSACTIONSId: a változást végrehajtó művelet kódja (egy TRANSACTIONS rekordra mutat) * FromSHIPPINGUNITId: a művelet által felhasznált SU rekord azonosítója (egy SHIPPINGUNITS rekordra mutat) * ToSHIPPINGUNITId: a művelet által létrehozott SU rekord azonosítója (egy SHIPPINGUNITS rekordra mutat) |
| PUTRX  A PACKAGINGUNITS tábla változásai | * Id: a rekord azonosítója * TRANSACTIONSId: a változást végrehajtó művelet kódja (egy TRANSACTIONS rekordra mutat) * FromPACKAGINGUNITId: a művelet által felhasznált PU rekord azonosítója (egy SHIPPINGUNITS rekordra mutat) * ToPACKAGINGUNITId: a művelet által létrehozott PU rekord azonosítója (egy SHIPPINGUNITS rekordra mutat) |
| MONITORDATA | * Id: a rekord azonosítója  (minden RECEIVING munkahelyhez pontosan egy rekord van a táblában, ami a pillanatnyi státuszt leíró  adatokat tartalmazza) * Instance: több azonos típusú (Type értéke) monitor adat lehet, az instance egy egyedi azonosító; ebben az esetben a munkahely megnevezése (csak egy munkahely-en mutatjuk be a folyamatot, de a táblába begenerálunk további munkahelyekre vonatkozó adatokat, így a monitor képernyőkön úgy fog látszani, mintha több munkahely működne párhuzamosan); pl. több bevételező munkahely van (Type), mindegyiknek van egy egyedi neve (Instance), és ennek megfelelő tartalommal rendelkező monitor adat rekord tartalmazza az adott munkahely pillanatnyi státuszhoz tartozó adatokat; egy bevételező munkahelynek a saját monitor adatai kiírásakor a saját típusára (Type) és egyedi nevére (Instance) kell hivatkoznia. * iType: a monitor adat típusa / munkahely típusa * TS: a monitor adatrekord utolsó változásának időbélyege * Content : a monitor adatok (xml formátumban)  |  |  | | --- | --- | | RECEIVING | * InSUId: a munkahelyen átmozgó SU azonosítója * InPN: az SU -ban levő cikk cikkszáma * SUQty: az SU töltő mennyisége | | REPACKING | * InSUId: a munkahelyen lévő SU azonosítója * InPN: az ISUId-ben levő cikk cikkszáma * OldSUQty: az ISUId átcsomagolás előtti töltő mennyisége * NewSUQty: az ISUId átcsomagolás utáni töltő mennyisége * InPUId: a munkahelyen lévő PU azonosítója * PUQty: a PU töltő mennyisége (=OldSUQty-NewSUQty) | | PUTOUT | * InPUId: a munkahelyen átmozgó PU azonosítója * InPN: az PU-ban levő cikk cikkszáma * PUQty: a PU töltő mennyisége | | KANBAN | * CellStatus: inaktív, szabad, foglalt/üres,… értékek * IPUId-loaded: a cellában levő PU azonosítója (teli cellák esetén van értéke, egyébként üres) * IPUId-expected: a cellába betárolandó PU azonosítója (foglalt/üres és foglalt/teli/NOK cellák esetén, egyébként üres) * PUQty: a cellában levő PU-ban, levő mennyiség (csak ha IPUId-loaded ki van töltve, egyébként üres)   Megjegyzés: ebben az esetben minden kan-ban cellához pontosan egy monitor adat tartozik (egy Instance egy kan-ban cella), ennek megfelelően az Instance értéke lehet pl. S1O2. |   Az xml adattartalom használatát a következő teszt t-sql script szemlélteti. Ennek alapján a fenti MONITORDATA táblát a WebMonitor program közvetlenül tudja használni.  use LearALM  drop table XmlDemo  create table XmlDemo (  Id int PRIMARY KEY,  Instance nvarchar(max),  iType nvarchar(max),  TS DateTime,  Content xml,  )  insert into Xmldemo (Id,Instance,iType,TS,Content)  values (1,'AS01','RECEIVING','2019.09.05 16:00:00','<document>  <InSUId>S111111111111</InSUId>  <InPN>P111111</InPN>  <SUQty>100</SUQty>  </document>')  insert into Xmldemo (Id,Instance,iType,TS,Content)  values (2,'AS02','RECEIVING','2019.09.05 17:00:00','<document>  <InSUId>S222222222</InSUId>  <InPN>P222222</InPN>  <SUQty>200</SUQty>  </document>')  Select \* from XmlDemo  Select  Id  ,Instance  ,iType  ,TS  ,Content.value('(/document/InSUId)[1]','nvarchar(max)') AS InSUId  ,Content.value('(/document/InPN)[1]','nvarchar(max)') AS InPN  ,Content.value('(/document/SUQty)[1]','nvarchar(max)') AS SUQty  from XmlDemo  where iType = 'RECEIVING' |

## Feladatkiosztás

|  |  |  |  |
| --- | --- | --- | --- |
| F1 | RedisEventHub fejlesztése | Görcsös Zoli | A RedisEventHub komponenst (az EventHubCore osztályt), ami az RedisEventHub kezelésének eszközeit tartalmazza, a végletekig leegyszerűsített használat érdekében ki kell egészíteni a következőkkel: |
| F2 | Log4ProIS-TRM megvalósításának előkészítése | Görcsös Zoli | ApplicationContainer plugin keretének elkészítése, RedisEventHub interfész használati minta beépítése, EntityFramework DBCOntext használati minta elkészítése, (ha kell), egy ivXXX beavatkozást feldolgozó metódus előkészítése |
| F3 | Teszt web alkalmazás és Log4ProIS-TRM web-POST akció elkészítése | Hajdu Albert | Teszteléséhez web alkalmazás és Log4ProIS-TRM modulhoz web-POST akció elkészítése |
| F4 | Log4ProIS-TRM beavatkozás feldolgozások elkészítése | Hajdu Albert | A Zoli által előkészített ApplicationContainer plugin-ban. Tesztelés ugyancsak a Zoli által készített web felületen keresztül. |
| F5 | Log4ProIS-TRM adatbázis terv véglegesítése és DBContext elkészítése | Vadász Valentin |  |
| F6 | ReelCheck módosítása | Görcsös Zoli | a jelenleg a külső rendszerek felé kialakított web-POST hívás interfész mellé egy RedisEventHub technikát használó interfész kialakítása azonos funkcionalitás mellett |
| F7 | Log4ProIS-PUTOUT és Log4ProIS-RECEIVING | Hajdu Albert | FÜZI-WRCV és FÜZI-WPUTOUT modulok elkészítése (RedisEventHub alkalmazása) |
| F8 | Log4ProIS-REPACKING | Hajdu Albert | web-POST akció készítése a Zoli által kialakított web alkalmazásban (benne RedisEventHub alkalmazása); |
| F9 | Log4ProIS-KANBAN | Vadász Valentin | Windows service elkészítése a fentiek szerint |
| F10 | VRH.Common bővítése | Schwirg László | ReturndictJSON típus elkészítése |
| F10 | WebForm definíciók, WebMonitor definíciók, Menü Definíciók | Schwirg László |  |

## F1 - RedisEventHub fejlesztése feladat (plusz info F3,F4,F7,F8,F9 feladatokhoz)

Ez a rész tartalmaz RedisEventHub használati mintát is mind a feladó/kliens, mind pedig a fogadó/szerver oldali fejlesztésekhez.

|  |  |
| --- | --- |
| **EventHubCore.StandardReg metódus elkészítése** | Használata a szerver oldalon:  string channelId = „CHANNELID”;  EventHubCore.StandardReg(channelId, StandardAction);  ami ekvivalens a jelenleg használható alábbival:  string channelId = „CHANNELID”;  EventHubCore.RegisterHandler<RedisPubSubChannel,  InterventionContract.StandardParameters,  Dictionary<string, string>>(channelId, **StandardAction**);  ahol  public class InterventionContract  {  public class StandardParameters  {  public string IVName { get; set; }  public Dictionary<string, string> Parameters { get; set; }  }  }  és  private Dictionary<string, string> **StandardAction** (string ivname, Dictionary<string, string> p)  {  var returndata = Dictionary<string, string>();  switch(ivname)  {  case „IVname1”:  returndata.Add(”RETPARNAME1”, ”X” + p[”PNAME1”]);  returndata.Add(”RETPARNAME2”, ”X” + p[”PNAME2”]);  break;  case „IVname2”:  returndata = ExecuteIVname2(p)  ;break;  default:…  **itt mit kell csinálni ahhoz, hogy a hívó oldalon (a StandardCall hívásban) kiváltódjon egy exception?**  }  return returndata;  }  Ennek a StandardAction metódusnak a megfelelő kitöltése lesz Albert feladata. |
| **EventHubCore.StandardCall metódus elkészítése** | Használata a kliens oldalon:  string channelId = „CHANNELID”;  string ivname = "IVname1";  var p = new Dictionary<string, string>();  p.Add(„PNAME1”,”pval1”);  p.Add(„PNAME2”,”pval2”);  TimeSpan ts = new TimeSpan(0, 0, 5);  Dictionary<string, string> r;  r = EventHubCore.StandardCall(channelId, ivname, p, ts);  string rp1 = r[”RETPARNAME1”]; // == ”XPNAME1”  string rp2 = r[”RETPARNAME2”]; // == ”XPNAME2”  ami ekvivalens a jelenleg használható alábbival:  string channelId = „CHANNELID”;  var ip = new InterventionContract.StandardParameters();  ip.IVName = "IVname1";  ip.Parameters = new Dictionary<string, string>();  ip.Parameters.Add(„PNAME1”,”pval1”);  ip.Parameters.Add(„PNAME2”,”pval2”);  TimeSpan ts = new TimeSpan(0, 0, 5);  Dictionary<string, string> r = EventHubCore.Call<RedisPubSubChannel, InterventionContract.StandardParameters, Dictionary<string, string>>( channelId, ip, ts);  string rp1 = r[”RETPARNAME1”]; // == ”XPNAME1”  string rp2 = r[”RETPARNAME2”]; // == ”XPNAME2”  Ezzel a **StandardCall** metódussal kell a sziget modulokban meghívni a visszakövetés modul lekérdezéseit, beavatkozásait és a monitor adatok rögzítésekor (Albert,Valentin). |

## F3 - Teszt web alkalmazás és Log4ProIS-TRM web-POST akció elkészítése

Egy web alkalmazás készítése, amibe a bemutatóhoz szükséges webes komponenseket fogjuk beépíteni. Nem tartalmaz semmilyen más modult/controllert, csak az alábbiakat. Az ebben levő akciókat egy külső web-alkalmazásból hívjuk meg, amely tartalmazza a menüt, a webform-ot, a webmonitort,….és egyéb általános célú modulokat.

A Home Controllerben készül egy POST akciót a visszakövetés modul teszteléséhez, aminek paraméterei: channel, ivname, timespan, PNAME1,PNAME2,…

channelid és ivname kötelező, timespan opcionális url paraméterek, a többi neve és értéke is tetszőleges, és lehetnek url paraméterek és dict paraméterek egyaránt.

Ez az akció elvégzi a következő hívást:

string channelId = [url paraméter: channelid értéke];

string ivname = [url paraméter: ivname értéke];

var p = new Dictionary<string, string>();

p.Add(”PNAME1”, [url paraméter: PNAME1 értéke]);

p.Add(”PNAME2”, [url paraméter: PNAME2 értéke]);

…

TimeSpan ts = new TimeSpan(0, 0, [url paraméter: timespan értéke, vagy alapértelmezés 10mp]);

Dictionary<string, string> r;

r = EventHubCore.StandardCall(channelId, ivname, p, ts);

Az akció a WebForm programmal való használhatóság érdekében egy JSON adatstrukturát ad vissza.

Ezt a VRH.Common modulba beépítjük, a neve ReturndictJSON, és a következőnek felel meg:

class ReturndictJSON

{

string name;

Dictionary<string,string> data;

}

A teszt hívások feladása a WebForm-mal történik, és a visszatérő adatokat is a WebForm jeleníti meg.