SZAKDOLGOZAT

Szegedi István

Debrecen

2010

Debreceni Egyetem

Informatikai Kar

Webáruházak szinkronizálásának módjai és azok összehasonlítása

Belső témavezető:

Dr. Adamkó Attila

egyetemi adjunktus

Külső témavezető:

Gál János

Készítette:

Szegedi István

programtervező informatikus (BSC)

Debrecen

2010

Tartalomjegyzék

1. Bevezetés

Dolgozatomban a webáruházak és az integrált vállalatirányítási rendszerek közötti szinkronizáció módszereit ismertetem és az adatátviteli módokat összehasonlítva, megvizsgálva azok előnyeit és hátrányait, kiválasztom a leghatékonyabb megoldást. Témámat a PHP szkript nyelven megírt weboldalakra szűkítem le, mivel ez az egyik leggyakrabban használt eszköz a honlap készítők körében.

Egy webáruház szinkronizálására különféle esetekben kerülhet sor. Előfordulhat szinkronizáció két vagy több webshop között, az e-kereskedelem ezen formája a Business-to-Business (B2B). Míg ha egy vállalat rendelkezik offline üzlettel és online üzlettel is, akkor egy raktárkészlet kezelő software és a webáruház között is létrejöhet szinkronizáció, ezt nevezzük Business-to-Consumer (B2C) –nek. Mivel egyik rendszernek sem szükséges ismernie a másik működését, elegendő a szinkronizálandó adatokat konfigurálni, ezért a két e-kereskedelmi formát együtt kezelem a szinkronizációs módszerek tekintetében.

A bevezetés után a második fejezetben az elméleti hátterét ismertetem a témának. Definiálom a webáruházat, bemutatom a vásárlási folyamat résztvevőit, amelyek az adatcsere során fontosak számunkra. Ezután kitérek a szinkronizációra, mint fogalomra, annak részeire, illetve arra, hogy milyen adatokat szinkronizálhatunk és ehhez milyen technológiákat használhatunk fel. A webszolgáltatások részben az XML jelölőnyelvet, a SOAP, WSDL és UDDI protokollokat mutatom be. A szinkronizációs módokat is ismertetem, amelyeknél tapasztalataim során megismert webáruházakból említek meg példákat. Ezen fejezet legvégén a szinkronizációs módokat hasonlítom össze a költség, a gyorsaság, a megbízhatóság és a biztonság alapján.

Szakdolgozatom gyakorlati részében egy Joomla alapú webáruházhoz készítettem egy szinkronizációs komponenst. A komponens képes a SOAP klienssel kommunikálni, és az érkező adatokat feldolgozni. A 3. fejezetben ezt a komponenst és működésének részletes folyamatát ismertetem a SOAP klienstől érkező hívástól a naplózásokon, vizsgálatokon és a tároláson keresztül a SOAP szerver által visszaadott válaszig. A fejezet végén a komponens két speciális eszközét is bemutatom, az egyik a felparaméterezett URL –en keresztül történő szinkronizáció lehetősége, a másik pedig a rendelések exportálása XML dokumentum létrehozásával.

A szinkronizálás, mint a mai webáruház rendszerek egyik központi témája, elengedhetetlen fogalommá vált a fejlesztők számára. A témával kapcsolatban még kevés publikáció jelent meg az utóbbi években, de a jövőben valószínűnek tartom, hogy erről a témáról fognak még megjelenni könyvek, dokumentumok. Addig is, remélem értékes információkat, megoldásokat oszthatok meg az olvasóval.

2. Elméleti háttér

Mindenekelőtt fontosnak tartom a szakdolgozatom témájához kapcsolódó alapfogalmak meghatározását. Ebben a fejezetben a webáruház fogalmát próbálom tisztázni az értékesítési folyamat paramétereivel együtt. Leírom a webáruház főbb jellemzőit és a vásárlási folyamat menetét. Később kitérek a szinkronizálás definiálására, milyen adatokat tudunk szinkronizálni, hogyan történhet a kommunikáció és milyen technológiákat használunk fel az adatcsere során. A fejezet végén ismertetem a szinkronizálási módszereket és eszközöket. Összehasonlítom az adatátviteli módokat, kiemelem azok előnyeit és hátrányait, illetve példákat hozok megvalósításukra.

2.1. Webáruház fogalma

Napjaink internet központú világában, a weboldalak egyik legnépszerűbb csoportjai a webáruházak. Fogyasztók milliói használják nap, mint nap arra, hogy a bevásárlást kényelmesen otthonról végezzék el. A webáruház fogalma szorosan összekapcsolódik az e-kereskedelem (e-commerce) fogalmával. Tulajdonképpen az e-kereskedelem egyik megvalósulási formája.

Az elektronikus kereskedelem fogalmát gyakran keverik az e-business fogalmával, ezért fontos, hogy meghatározzuk mit is jelent e két fogalom. Az e-business magában foglal minden elektronikus tranzakciót, átutalást, vásárlást és minden olyan tevékenységet, amit elektronikus kommunikáció segítségével hajtunk végre. Az előbbi megfogalmazásból látható, hogy az elektronikus kereskedelem része az e-businessnek. Az elektronikus kereskedelem a termékek és szolgáltatások online értékesítését, az elektronikus beszerzést, a beszállítói és partneri kapcsolatok fenntartását jelenti.

Az e-business területet a következőképpen szokták osztályozni:

- Business to Business **(B2B):** üzleti partnerek közötti elektronikus kapcsolat
- Business to Consumer **(B2C):** internetes kiskereskedelem, fogyasztóknak, végfelhasználóknak szóló marketingtevékenység
- Business to Administration (**B2A):** vállalatok és a kormányzat közötti tranzakciók

- Consumer to Administration **(C2A):** lakosság és a kormányzat, hivatalos szervek közötti kapcsolat
- Consumer to Consumer **(C2C):** fogyasztók közötti kapcsolat, aukciók
- Business to Employee **(B2E):** vállalat és alkalmazottak közötti kapcsolat

A webáruházakat érintő területeket, a Business to Business (B2B) és a Business to Consumer (B2C) osztályokat részletesebben is meg kell vizsgálnunk.

Business to Consumer (B2C):

Az üzlet és a fogyasztók közötti kereskedelem a legnépszerűbb és legismertebb forma ezen a területen. Az online boltok, e-bevásárló központok, e-bankok, e-bemutatótermek, e-utazási irodák, e-szállodafoglalás tartozik ebbe a csoportba. Annak ellenére, hogy ez a legnépszerűbb formája az e-kereskedelemnek, a B2B tranzakciók összértéke jóval magasabb, mint a B2C tranzakcióké. Ez nem jelenti azt, hogy kevesebb tranzakció történik egy B2C területen.

Business to Business (B2B):

Az e-kereskedelem e formája a vállalatok közötti elektronikus úton történő kapcsolattartást és üzletkötést jelenti. A B2B alapú kereskedelem nagyobb múltra tekint vissza, mint a B2C alapú. A korai kereskedelemben a vállalatok között EDI (Electronic Data Interchange), azaz elektronikus adatcsere zajlott le. Informatikus szemmel nézve a B2B és a B2C közötti legfontosabb különbség, hogy a B2B esetében két szervergép kommunikál egymással.

Dolgozatomban a B2B és B2C formákat együtt fogom kezelni, mivel mindkét esetben ugyanazok a szinkronizációs módszerek alkalmazhatóak. A két formánál a külső rendszerrel történő szinkronizáció jellege szinte teljesen megegyezik. Így tehát nem teszek különbséget B2B és B2C webshop között. Webáruház alatt, egy általános fogalomként ismert weboldalt értek, ahol termékek vagy szolgáltatások értékesítése folyik. Későbbiekben a webáruház szinonimájaként az internetes bolt, webshop, online bolt, áruház vagy online áruház kifejezéseket is használom.

Érdemes megvizsgálni a webáruház azon egységeit, melyek fontosak lehetnek a szinkronizálás során. Ezek lesznek az áruházban vásárolható termékek, a vásárlók

rendelései és a felhasználók adatai. A következő fejezetekben egy konkrét webáruház elemeit ismertetem. Természetesen ettől eltérő szerkezetű is lehet egy webshop, de dolgozatomban csak az általam bemutatott felépítéssel foglalkozok.

2.1.1. Termékek a webáruházban

A webáruházakban forgalmazott termékek nem feltétlenül jelentenek fizikailag megjelenő árukat. Lehet egy konkrét termék, amit meg lehet rendelni és le lehet szállítani. Ezen kívül az áruházban értékesíthetnek egy szolgáltatást is, például fűnyírás a kertben, kerttakarítás stb. Az egyszerűség és a könnyebb megértés érdekében olyan terméket vizsgálok, amely egy konkrét, fizikailag létező áru.

Az online vásárlás legfontosabb elemei tehát a termékek, melyek ágazatonként nagyon különbözőek lehetnek. Általában tulajdonságaikat többféle paraméterrel adhatjuk meg. Az alapvető adatai egy terméknek az ár, a terméknév, a termékleírás és a cikkszám. Ma már a webshopokban többnyire minden termékhez kapcsolódik egy vagy több termékkép is. Ezen kívül lehetnek további speciális tulajdonságai, mint például a gyártó, súly, méretek, webcím.

Nemcsak olyan paraméterei léteznek egy árunak, amelyek a fogyasztók számára is láthatóak. Vannak rejtett beállítások is, mint a láthatóság (megjelenik-e a frontend-en), prioritás (rendezésnél fontos), mikortól kapható, kiemelt-e a termék, vagy a raktárkészlet, ami a szinkronizálás leggyakoribb eleme. Speciális webáruházak esetében egyéb paraméterekről is beszélhetünk. Egy könyváruházban van szerzője, kiadója és kiadási ideje a könyvünknek, egy motorolajakat forgalmazó cég megadja, hogy benzines vagy dieseles autóhoz ajánlja a terméket, illetve annak kiszerelését. Egy ruházati boltban egy pulóvernek vannak méretei és színt is lehet választani.

Láthatjuk tehát, hogy a paraméterek korlátlanok lehetnek, amelyre a szinkronizálás során is fel kell készülni.

2.1.2. Rendelések

Az elektronikus kereskedelem során a fogyasztók a kiválasztott termékeket megrendelhetik, de hogyan jön létre a vásárlás? Internetes boltunkban a vásárlás tulajdonképpen a rendelés visszaigazolásával jön létre. Ez a folyamat úgy történik, hogy

a vevő kiválaszt egy terméket, amit a kosárba tesz, ezután a legtöbb bolt esetében regisztráció szükséges a további lépésekhez, mellyel létrejöhet a rendelés. A regisztrációtól függetlenül a rendeléshez meg kell adnunk a számlázási és szállítási címünket, a szállítás módját és a fizetés módját. Ezeket az adatokat nevezzük megrendelői információknak, mivel a vásárló egyéni igénye alapján jön létre.

A rendelés tartalmazza továbbá a kiválasztott termék vagy termékek szükséges adatait, a végösszeget, az akciót, az adó mértékét. Ezeket nevezzük a vásárolt termékhez kapcsolódó adatoknak.

Szinkronizálás során a legtöbbször módosításra kerülő, rendeléshez kapcsolódó adat a rendelés státusza. A vásárló miután elküldi a rendelést „nyers" státuszként tárolódik el. A boltvezető, amint észlelte a rendelést egyes boltoknál megadhatja a „feldolgozva" státuszt, mellyel informálja a vevőt, hogy a boltvezető elfogadta rendelését. Ezután a „szállítva" majd a „teljesítve" státuszokat állíthatja be az adminisztrátor. Ez az adat, melyet email értesítésben kap meg, fontos információt jelez a vevő számára a rendelés állapotáról.

2.1.3. Vásárló adatai

Mint már említettem a vevő rendelés során megadhatja adatait. A webáruházakban általában regisztrációhoz kötött a vásárlás, de a rendelés során lehetőség van a korábban regisztrált adatok módosítására. A regisztrációnál a vásárló megad egy felhasználónevet, a belépéshez egy jelszót, email címet, esetleg cégnevet és egyéb elérhetőségi adatokat, mint a telefonszám vagy faxszám és a lakcím. Fontos, hogy a regisztrációnál megadott lakcím és adatok lesznek később a számlázási adatok, míg a szállítási adatokat külön kell megadni. Egy felhasználónak akár több szállítási címe is lehet.

A felhasználók adatait olyan esetben szoktuk szinkronizálni külső rendszerekkel, ha például egy hírlevél küldő rendszerbe szeretnénk átvinni az adatokat, vagy ha a cégünk ügyfeleit szeretnénk összegyűjteni az ügyviteli vagy számlázási rendszerünkben.

2.2. A szinkronizáció fogalma a webáruházak világában

A szinkronizáció azt jelenti a számítástechnikában, hogy két egymással kapcsolatban lévő rendszer között, ha a tárolt adatokban változás történik, akkor mindkét rendszerben módosulnak az adatok. A kommunikáció a két fél között lehet automatikus vagy ütemezett, melynek eredményeként a két rendszer adatbázisának tartalma megegyező lesz. Webáruházak esetében akkor beszélhetünk szinkronizációról, ha a webshop adatbázisa és az ügyviteli rendszer vagy egyéb integrált vállalatirányítási rendszer (ERP - Enterprise Resource Planning), vagy alkalmazás között történik az adatcsere. Ezen fejezetet az elektronikus adatcsere fejlődésével kezdem, majd kifejtem a szinkronizáció jelentőségét, végül háromféle szempontból is osztályozom az adatátvitelt. Természetesen ezt a témát is a webáruházakra szűkítem le.

2.2.1. Az elektronikus adatcsere fejlődése

Az e-kereskedelem kezdeti elektronikus adatcserét szolgáló technológiája az EDI (Electronic Data Interchange) volt, amely két vállalat között meghatározott adatforgalmat biztosított. Az EDI definíciója: „Strukturált adatok szabványos elektronikus cseréje kettő vagy több, előzetesen egyeztetett üzenettovábbító szabványt használó számítógéprendszer között.”[1] A nagyvállalatok körében terjedt el, akik így biztosították gyors és megbízható adatcseréjüket a beszállítóikkal. Az EDI az államigazgatásban és a kereskedelemben a legnépszerűbb adatátviteli eszköz.

Az EDI rendszerben a küldő és fogadó számítógépek közötti adatcsere során, tudnunk kell hitelesíteni az átmenő adatokat, amelyek a megfelelő szabványok szerint néznek ki és azonosítanunk kell a két felet. Ezeket a formai és funkcionális szabályokat, szabványokat adja meg az EDIFACT (Electronic Data Interchange For Administration, Commerce and Transport). Ezt a szabványrendszert az ENSZ szervezetei dolgozták ki.

Egy EDI rendszerben három folyamatról beszélhetünk. Először az adatokat olyan formátumra alakítjuk, mely továbbításra alkalmas. Ezután az üzenetet továbbítjuk, vagy fogadjuk, majd a harmadik folyamat során az üzeneteket visszaalakítjuk olyan formára, amelyet fel tudunk dolgozni.

[1] Miniszterelnöki Hivatal - Elektronikus adatcsere alkalmazása a kormányzatban

Később jött a web-EDI névre keresztelt megoldás, mely a vállalatok közötti adatcsere során már az XML technológiát is felhasználta. Itt az adatátviteli szabványrendszer már a HTTP volt. Mára azonban ezeket a technológiákat felváltotta egy új lehetőség, a webszolgáltatások.

A webszolgáltatások olyan protokollok és szabványok gyűjteménye, amelyek az elektronikus adatcsere megvalósítását teszik lehetővé. A legelterjedtebb webszolgáltatás protokollok a SOAP, WSDL és UDDI. Ezeket a protokollokat használhatjuk fel a webáruházak és ügyviteli rendszerek közötti szinkronizáció során is. Dolgozatom későbbi fejezeteiben részletesen ismertetem a webszolgáltatások ezen protokolljait.

2.2.2. A szinkronizáció jelentősége

A mai világban nagy jelentősége van az adat szinkronizációnak. Gondoljunk csak bele, hogy egyre többen választják az internetes vásárlást és egyre növekszik a webáruházak száma a világhálón. Mivel egyre többen használják a vásárlás e formáját, ezért egyre több embernek és igénynek kell megfelelni.

Miért nőtt meg az interneten vásárlók száma? Ennek a növekedésnek a hátterében a kényelem és az egyszerűség áll. Míg egy offline vásárlás esetén, az általunk kiválasztott terméket több üzletet végigjárva találhatjuk meg, ezzel szemben az online piac egyik nagy előnye, hogy otthonról a keresők segítségével pillanatok alatt eljuthatunk a kívánt termékhez és néhány perc alatt megrendelhetjük.

Vannak olyan webshopok, ahol a termékek száma nem haladja meg az 50-100 darabot, mint például egy kisebb ruhaüzlet, ami az offline boltját egészíti ki online értékesítéssel, vagy egy mezőgazdasági gépeket forgalmazó cég, amely kevesebb, mint 50 különböző típusú gépet forgalmaz. Ezeknél az áruházaknál ritkábban változik a raktárkészlet vagy a termékek ára, ezért ritkábban van szükség ezek frissítésére. Azonban azokban a webáruházakban, ahol gyakoriak az akciók és hetente változik a termékkészlet, szükség van szinkronizációs megoldásra.

Az értékesítési folyamat során az ügyfél igényeit kell szem előtt tartani. Ha egy olyan áruházban találja meg a kívánt terméket a vásárló, ahol nincsenek frissítve annak adatai és ezért esetleg túl drágának találja - és miért ne találná az 1 hónappal ezelőtti árat - akkor tovább áll és máshol próbálja megvásárolni, ezáltal az üzlet veszteséget szenved.

Bizalomépítő lehet egy olyan webshop, ahol feltüntetjük a raktárkészletet, így a vevő láthatja a pontos árakat, akciókat és mindig naprakész információkat kaphat.

Egy másik megközelítésben egy nagyvállalatot vizsgálok meg, melynek üzleti alkalmazásai, számlázó programja és ügyviteli rendszere egy komplex vállalatirányítási rendszer részei. Ilyenkor a cég a termékek adatait az ügyviteli rendszerében tárolja és ahhoz, hogy a webáruházban is mindig ezek az adatok jelenjenek meg szükség van szinkronizációra. Mennyi energiát és időt spórolhat meg egy ilyen cég, ha létrehozzák a vállalatirányítási rendszer és a webshop között a szinkronizációt. Ha manuálisan kellene feltölteni az adatokat és frissíteni a termékeket, külön embereket kellene foglalkoztatniuk, ami pénzkidobás lenne. Nem beszélve arról, hogy senki sem szeret duplán dolgozni. Általános igény, hogy ha egy terméket feltöltöttem az ügyviteli rendszer adatbázisába, akkor ne kelljen még egyszer ugyanazokat az adatokat a webshop adatbázisába is feltölteni. A szinkronizáció jelentősége tehát az automatizálásban rejlik, amely többnyire minden esetben megvalósítható.

2.2.3. Szinkronizáció osztályozása adatátvitel iránya szerint

Az adatátvitel iránya szerint webáruházunk szempontjából két osztályba sorolhatjuk a szinkronizációt. Az egyik osztály az exportálás, amikor az online áruház adatbázisában lévő adatokat mentjük át egy másik külső rendszerbe. A másik, amikor a külső rendszerből érkezik az adat a webshop adatbázisába, ez az importálás. Röviden tehát exportálás során mi küldjük, importálás során pedig mi fogadjuk az adatot. Nézzük kicsit részletesebben a két részt:

Exportálás

Általában abban az esetben van szükség a webáruházunkban tárolt adatok külső rendszer felé történő szinkronizálására, ha a külső rendszer például egy számlázó program, vagy egy hírlevél küldő szoftver. Leggyakrabban az oldalunkra beregisztrált userek adatait exportáljuk, beleértve a hírlevelünkre feliratkozott látogatók információit is. A másik leggyakoribb kimentett adat a rendelés, amely fontos információt szolgáltat például a számlázó programoknak vagy ügyviteli rendszereknek. A termékek paraméterei közül egyedül a raktárkészlet lehet fontos számunkra. Ha például 5 darabot rendelnek egy termékből a webes felületen, akkor értesítenünk kell a raktárkezelő

rendszert, hogy a webshopban csökkent a raktárkészletünk, amely szintén exportálási folyamat.

Importálás

Webáruházak esetében sokkal gyakrabban használunk importálást. Gyakorlatilag hívhatnánk adatfeltöltésnek is, mivel az ügyviteli rendszerünk adatbázisában tárolt termékek árát, paramétereit, láthatóságát, raktárkészletét vagy a termékkategóriák szerkezetét visszük át internetes boltunk adatbázisába. A legtöbb üzlet hetente kap új termékeket, amit fel kell tölteni a webshopba is, ilyenkor a termékeket importáljuk az adatbázisba. Kategóriák importálására akkor lehet szükség, ha nagy terjedelmű és több szintű kategóriafa található a boltban. Ilyenkor célszerű az adatbázisba importálni a termékkategóriákat. Rendelések esetében a rendelés állapota változhat az ügyviteli rendszerben. Tehát, ha a webshop adatbázisából kiexportált megrendelés státuszát az ügyviteli rendszerben az üzletvezető módosítja, akkor ezt a változtatást vissza kell töltenünk.

2.2.4. Szinkronizáció osztályozása az idő függvényében

Az idő függvényében is két részre oszthatjuk a szinkronizációt. Ha az adatátvitel a két fél között valós időben (real-time) történik, akkor szinkron átvitelről, ha viszont a kommunikáció nem egy időben történik, vagy ütemezett módon, akkor aszinkron átvitelről beszélhetünk.

Szinkron adatátvitel

Azonnali kommunikációt jelent, például ha egy vásárló megnézi egy termék adatlapját, akkor a webáruház elküld egy kérést a külső rendszer felé, hogy mondja meg neki a raktárkészletet vagy a termék aktuális árát. Ez a rendszer válaszol az üzenetre és ezt az információt jeleníti meg a webshop a felhasználó számára. Mindezt a másodperc töredéke alatt teszi. Ez a fajta adatátvitel mindkét fél egyidejű közreműködését igényli.

Előnye, hogy a felhasználó mindig naprakész adatokat lát az oldalon. Hátránya viszont, hogy bonyolultabb a megvalósítása és éppen ezért drágább is az informatikai rendszer kiépítése. Ezenkívül jóval több kezdeti (és gyakran folyamatos) költséggel is jár, mivel nem csak a fejlesztés problémásabb az ilyen rendszereknél, de gyakran egyéb infrastrukturális beruházással is járna (például külön szervergép vásárlásával a

webáruház számára, vagy elegendő sávszélesség biztosítása). Ilyenkor ugyanis sokkal több adat áramlik a webshop és a külső rendszer között, így a terhelhetősége is jóval korlátozottabb.

Ezen adatátvitelt olyan vállalatok alkalmazzák, amelyeknek egyrészről létezik az az erőforrás, amellyel megoldható a valós idejű kommunikáció, más részről olyan ügyfelekkel, vásárlókkal állnak kapcsolatban, akiknek fontos, hogy a feltüntetett adatok a valóságnak megfelelőek legyenek és azonnali rendelés esetén, azonnal kézhez kapják a termékeket.

Aszinkron adatátvitel

Ilyenkor a webáruház és a külső rendszer között adott időközönként történik kommunikáció. Ez történhet óránként, naponta, hetente vagy esetleg egy adott esemény bekövetkezésekor, például ha megnyomunk egy gombot, vagy egy felparaméterezett URL-t hívunk meg. Ekkor a legutolsó kapcsolat óta létrejött változások kerülnek módosításra. Előnye, hogy könnyebben megvalósítható, mint egy real-time rendszer, hátránya viszont, hogy mivel nem valós idejű, ezért az adatok sem mindig teljesen frissek. A frissítések gyakoriságának növelésével csökkenthető a hibaszázalék, de teljesen nem tüntethető el. A felhasználó szempontjából gyakorlatilag észre sem vehető a különbség, amennyiben a frissítések gyakoriságát minimálisra, néhány percre csökkentjük. Természetesen minél gyakoribb az adatcsere, annál nagyobb erőforrást igényel.

Szinkron és aszinkron adatátvitel összehasonlítása

	Szinkron adatátvitel	Aszinkron adatátvitel
Kommunikáció	Valós időben történik	Adott fix időközönként
Előnye	Mindig naprakész adatok	Egyszerűbben és olcsóbban kivitelezhető
Hátránya a vevő szemszögéből	Normál forgalomnál nincs, magas forgalom esetében belassíthatja a rendszer működését	Elavult adatok megjelenítése, melyekből származó problémák minimálisra csökkenthetőek
Hátránya az eladó szemszögéből	Nincs, teljesen automatizálható	Nincs, teljesen automatizálható
Költsége	Magasabb kezdeti beruházás és általában magasabb folyamatos költség	Általában alacsonyabb kezdeti beruházás és gyakorlatilag nulla folyamatos költség

2.2.5. Szinkronizáció osztályozása az automatizáltság foka szerint

A harmadik csoportosítás az automatizáltság foka szerint történik. Ez alapján megkülönböztetünk manuális, félautomatikus és automatikus szinkronizációt.

Manuális szinkronizációnak nevezzük azt a megoldást, amikor külső rendszer alapvetően nincs felkészülve semmilyen külső kommunikációra, de lehet bele adatokat importálni, illetve belőle adatokat kiexportálni. Ilyenkor a külső rendszerben lévő adatokat egy fájlba (jellemzően CSV, XLS vagy XML) exportálhatjuk, amelyet egy-az-egyben a webáruház adatbázisába importálunk. Hátránya, hogy a webáruháznak alkalmazkodnia kell a külső rendszerhez, mely speciális esetekben módosítást igényel a webáruház rendszerében. Előnye viszont, hogy gyakorlatilag minden ügyviteli rendszer képes az exportálásra.

Félautomatikus szinkronizációnak nevezzük az olyan rendszereket, amelyek képesek egymással kommunikálni, de ezt szigorúan csak emberi parancsra teszik. Ilyenek gyakran a számlázó programok, melyekben egy gomb lenyomásával le lehet tölteni a friss rendeléseket.

Automatikus szinkronizáció egy olyan megoldás, amelyhez semmilyen emberi beavatkozás nem szükséges a hétköznapi beállításokhoz, csak egyedi fejlesztések esetén kell a rendszerbe belenyúlni. A real-time rendszerek mind automatikusak, az aszinkron rendszerek esetében fordulhat elő a manuális vagy félautomatikus működés. Alapvetően célszerű automatikusra készíteni azokat a folyamatokat, amely nem igényel emberi beavatkozást. Például a termékadatok automatikusan frissülhetnek, viszont a megrendelés státuszát úgyis az üzletvezető fogja módosítani.

2.3. Szinkronizált adatok

Az előző pontban ismertettem az exportálás és importálás fogalmát, illetve bemutattam mely esetekben használjuk őket. Most azokat az adatokat gyűjtöm össze, amelyeket webáruházak esetében exportálni és importálni szoktunk. Az adatokat pontokba szedve részletezem, és felsorolom, hogy milyen paraméterek szükségesek vagy opcionálisak az adatátvitel során. A felsorolásban szereplő adatok fájlok esetében használatosak (XML, CSV, XLS). Az adatok reprezentálása többféle módon történhet, a sok megvalósítás közül egyet választottam ki.

2.3.1. Kategóriák szinkronizálása

A termékkategóriák szinkronizálására, mint már a korábbiakban említettem, importálás során van szükség. Kisméretű kategóriafa esetében a kézi feltöltés is könnyen megoldható, de egy nagyobb méretű esetén már elengedhetetlen az importálás megvalósítása.

Fájlban szereplő adatok:

- ✓ SzuloID (szülőkategória azonosítója, amely 0, ha főkategóriáról van szó)
- ✓ ID (azonosító)
- ✓ Name (név)
- ✓ CategoryImage (fájlnév) – opcionális
- ✓ Description (leírás) – opcionális

Egyedi webáruházak esetében létezhetnek egyéb opcionális paraméterek is, melyek közül a legáltalánosabbakat vettem figyelembe.

2.3.2. Termékadatok szinkronizálása

A webáruház legfontosabb részei a termékek. Egy termék lehet akár néhány paraméteres, de lehet közel 100 paraméteres is. Például egy cipő esetében lehet márkát, méretet és színt megadni, míg egy mobiltelefon vagy laptop esetében sokkal több paramétert kell megadni. Természetesen minden terméknek vannak kötött jellemzői, amelyek minden boltnál léteznek és többségüknek létezniük is kell. Ezek a termékek alapadatai, a termék árai, láthatósága és raktárkészlete.

- **Termék alapadatai**

Ide tartoznak azok az adatok, melyek egy általános webshopban a termékhez kapcsolódnak. Van közöttük opcionális paraméter, amely minden boltnál létezik, de lehet, hogy nem használják.

Fájlban szereplő adatok:

- ✓ SKU (cikkszám)
- ✓ Name (termék neve)
- ✓ CategoryID (kategória azonosítók ; -vel felsorolva)
- ✓ ShortDescription (rövid leírás) – opcionális
- ✓ LongDescription (hosszú leírás) – opcionális
- ✓ RegistrationDate (létrehozás dátuma)
- ✓ Price1 (nettó ár)
- ✓ Vat (áfa)
- ✓ Discount (akciós ár) – opcionális
- ✓ Unit (mértékegység)
- ✓ Stock (Készlet db)
- ✓ Image (Fájl név) – opcionális

- **Termék ár**

Bizonyos áruházakban többfajta árképzés létezik. Vannak vásárlói csoportokhoz, vagy kedvezményekhez kötött árak. Egy akció esetében, az akció időtartamát, vagyis az akció kezdetét és végét is megadhatjuk. Ilyenkor a termék frissítése során csak az árakat kell figyelembe venni, melyek egy külön adatbázis táblában tárolódnak.

Fájlban szereplő adatok:

- ✓ SKU (cikkszám, ez alapján kapcsolódik a termékhez az ár)
- ✓ Price1 (nettó ár, alapértelmezett user csoport esetén)
- ✓ Price2 (nettó ár, következő user csoport esetén)
- ✓ Price3 (nettó ár, következő user csoport esetén)
- ✓ Price4 (nettó ár, következő user csoport esetén)
- ✓ Price5 (nettó ár, következő user csoport esetén)
- ✓ Vat (áfa)
- ✓ Discount (akciós ár)
- ✓ Akció kezdete (Formátum: YYYY-MM-DD pl: 2010-03-01)
- ✓ Akció vége (Formátum: YYYY-MM-DD pl: 2010-03-09)

- **Termék raktárkészlete**

Korábban már említettem, hogy a raktárkészlet kezelése miért és mennyire fontos tényező a szinkronizációnál. Mind exportálás és mind importálás esetén előfordulhat, hogy kizárólag csak a raktárkészlet áll az adatcsere középpontjában.

Fájlban szereplő adatok:

- ✓ SKU (cikkszám)
- ✓ Stock (Készlet db)

- **Termék láthatóság**

A „Termékek a webáruházban” fejezetben már említést tettem a láthatóságról, ami azt jelenti, hogy az adatbázisban szereplő termékeket, ha nem szeretnénk megjeleníteni a látogatók számára, akkor a láthatóságát kikapcsoljuk. Gyakran sor kerülhet erre, például ha érkezik egy új kollekció az üzletbe és azokat szeretnénk megjeleníteni az áruházban, a régi termékeket pedig egy rövid ideig láthatatlanná tesszük. Ehhez elegendő a láthatósági paramétert frissíteni.

Fájlban szereplő adatok:

- ✓ SKU (cikkszám)
- ✓ Publish (Láthatóság Y/N)

2.3.3. Rendelések szinkronizálása

Talán a leggyakoribb igény a számla kiállítása, amelyhez a rendelési adatokra van szükségünk. A rendelések szinkronizálásakor ketté kell bontani a folyamatot. Amikor a webshop adatbázisában létrejött rendelést szeretnénk átvinni az ügyviteli rendszerünkbe, akkor minden adatra szükségünk van, mely a rendeléshez tartozik (exportálás). Amikor viszont az ügyviteli rendszerben módosítottuk a rendelés státuszát, akkor elegendő a státuszt frissíteni a weben (importálás).

Exportálás esetén fájlban szereplő adatok:

Mivel túl sok adatot tartalmaz ez a fájl, ezért nem térnék ki a részletes ismertetésre.

- ✓ Vásárolt termékek adatai
- ✓ A vásárló számlázási címe
- ✓ A vásárló szállítási címe
- ✓ A megrendeléshez kapcsolódó adatok (rendelés végösszege, áfa, akció, adó, státusz, megjegyzés, szállítási költség és típus, fizetési mód és költsége, stb.)

Importálás esetén a fájlban szereplő adatok:

A rendelési állapot módosításáról értesíteni kell a vásárlót, ezért van egy mail paraméter, melynek értéke akkor 1, ha még nem küldte ki a rendszer a státuszváltásról szóló értesítőt, különben 0.

- ✓ OrderID (Rendelés azonosító.)
- ✓ OrderState (Rendelési állapot. Mivel a webshopban a rendelési állapot külön táblában tárolódik, ezért vagy a rendelési állapot ID-ja vagy betűjele.)
- ✓ Mail (Ha 1 –re van állítva még nem volt kiküldve az értesítő a státuszváltásról.)

2.3.4. Felhasználók szinkronizálása

Általában a felhasználók adatait szoktuk exportálni külső rendszerekbe, de előfordulhat, hogy egy vállalat belső rendszerében egy kiterjedt vásárlói adatbázis található, amit az offline értékesítés vagy marketing során gyűjtöttek össze. Ezt az adatbázist a cég szeretné importálni újonnan beindított, vagy már létező webáruház rendszerébe. Ha az ügyviteli rendszerbe szeretnénk összegyűjteni a webshopunkba regisztrált felhasználókat, akkor exportáljuk az adatokat. Mind a két esetben ugyanazokat a

paramétereket kell megadni. Előfordulhat, hogy a webáruházunkhoz egy hírlevél vagy egy fórum komponens van telepítve, ahol a hírlevélre vagy a hozzászólások figyelésére fel lehet iratkozni. A feliratkozást általában ezek a komponensek a saját adatbázis táblájukban tárolják. Természetesen ilyenkor kevesebb adatot kell a fájlban megadni.

- **Felhasználói adatok**

Mivel beszélhetünk számlázási és szállítási adatokról, meg kell különböztetni ezeket, erre egy külön azonosító szolgál. Ezenkívül a vásárlói csoportot is meg kell határozni, hogy hova tartozik a vásárló. A többi adat általános és egyértelmű.

Fájlban szereplő adatok:

- ✓ UserID (felhasználó azonosítója)
- ✓ FirstName (vezetéknév)
- ✓ LastName (keresztnév)
- ✓ Company (cégnév)
- ✓ Phone (telefonszám)
- ✓ Fax (faxszám)
- ✓ Email (email cím)
- ✓ Zip (irányítószám)
- ✓ City (város)
- ✓ Address (lakcím)
- ✓ RegistrationDate (létrehozás dátuma)
- ✓ ShopperGroupID (vásárlói csoport azonosítója)
- ✓ AddressType (szállítási vagy számlázási információ)

- **Feliratkozók adatai**

Egy fórum, vagy blog rendszer esetleg a hírlevél esetén beszélhetünk feliratkozókról. Manapság mivel az internetre egyre jobban jellemző a WEB 2.0, ami arról szól, hogy a felhasználói közösségek szerkesztik az interneten az adatokat, egyre több webáruházba integrálnak ilyen komponenseket. A feliratkozás mellett figyelembe kell vennünk azokat, akik le is akarnak iratkozni. Erre egy státusz paraméter szolgál, melynek értéke 1, ha feliratkozott és 0, ha leiratkozott. Néhány komponens külön letárolja a teljes nevet,

melyet figyelembe vehetünk, de megoldhatjuk úgy is, hogy később a vezetéknévből és keresztnévből konkatenációval létrehozzuk.

Fájlban szereplő adatok:

- ✓ FullName (teljes név)
- ✓ FirstName (vezetéknév)
- ✓ LastName (keresztnév)
- ✓ Mail (e-mail cím)
- ✓ Status (feliratkozás állapota)

2.4. Adatátvitelhez felhasználható technológiák

Szakdolgozatom ezen fejezetében a szinkronizáció során felhasználható technológiákat ismertetem. Korábban már említettem, hogy mennyire fontos az automatizálási folyamat a webáruház és a külső rendszer között. Az internet adta lehetőségeket kihasználva az üzleti szférában is egyre inkább jellemző az alkalmazások, szolgáltatások közötti automatikus kommunikáció. Egyre elterjedtebb, hogy két rendszer között úgy jön létre a kapcsolat, hogy az egyik rendszer kiajánlja az információt a másik számára webszolgáltatásokon keresztül. Mit is jelentenek a webszolgáltatások?

2.4.1. Webszolgáltatások bemutatása

„A webszolgáltatások világa egy új webmodell világának mondható. Olyan szabványok halmazáról van szó, amelyek szolgáltatás-orientált, komponens alapú alkalmazásokat írnak le. A webszolgáltatások jól definiált interfésszel rendelkező, az alkalmazások által a weben keresztül könnyen elérhető funkciókat tartalmaznak.” [2]

A webszolgáltatások szabványosításával a W3C szervezet, vagyis a World Wide Web Consortium foglalkozik. A webszolgáltatások a HTTP-n, vagyis a 80-as porton keresztül kommunikálnak XML üzenetek segítségével. Az XML a webszolgáltatások nyílt szabványai közül az elsődleges szabvány, mely egy platformfüggetlen és programnyelv független leíró nyelv. Ez a nyelv határozza meg a kliens-szerver alapú rendszer közötti kommunikációt.

[2] Gottdank Tibor - Webszolgáltatások

Két fajta megközelítés létezik a webszolgáltatások architektúrájára. Az egyik amikor három szereplőt különböztetünk meg, a másik amikor rétegesen épülnek egymásra a protokollok. Utóbbi esetben négy fő rétegről beszélhetünk.

A webszolgáltatások szereplői:

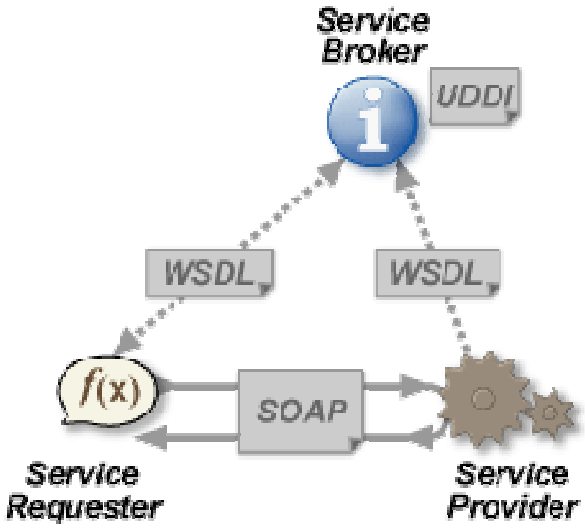
- **Szolgáltató:**
  Webszolgáltatást nyújt, közzéteszi a nyilvántartó számára a szolgáltatás-leírásokat. A szolgáltató felelős a létrehozásért, a publikálásért és az igénylőktől érkező webszolgáltatás-hívások fogadásáért.
- **Szolgáltatás igénylő:**
  A webszolgáltatás valamelyik felhasználója. A nyilvántartóban keres szolgáltatásokat. Felelős azért, hogy a nyilvántartóban megtalálják a szolgáltatás-leírásokat és ezek használatáért.
- **Szolgáltatás nyilvántartó:**
  Ez tulajdonképpen szolgáltatások gyűjteménye, egy nyilvántartás. A fejlesztők ebbe a gyűjteménybe publikálják szolgáltatásaikat, illetve itt keresnek a létező szolgáltatások között.

A webszolgáltatások rétegei:

- **Szolgáltatás átvitel:**
  Ez a réteg felel az üzenetek szállításáért. Leggyakrabban a már említett HTTP (Hypertext Transfer Protocol) –t használja, de az SMTP (Simple Mail Transfer Protocol) és az FTP (File Transfer Protocol) is használható.
- **XML alapú üzenetkezelés:**
  Az XML a protokollok közötti átjárhatóságot teszi lehetővé azzal, hogy közös, mindenki számára elérhető szabványt biztosít. A kommunikáció minden résztvevője tudja értelmezni. Ez a réteg jelenleg az XML-RPC-t és a SOAP (Simple Object Access Protocol) – t tartalmazza.
- **Szolgáltatás leírás:**
  Gyakorlatilag ez a réteg és ezen belül a WSDL (Web Service Definition Language) protokoll gondoskodik a szolgáltatások leírásáról.

- **Szolgáltatás felkutatása:**

  A közös szolgáltatás nyilvántartóban UDDI (Universal Description, Discovery, and Integration) –n keresztül történik a szolgáltatások kezelése, publikálása és keresése.



1. ábra *Webszolgáltatások felépítése*

A webszolgáltatások előnyei és hátrányai.

Előnyök	Hátrányok
Együttműködést biztosítanak a különböző platformokon futó alkalmazások között.	Nem képesek a tranzakció kezelést megoldani, vagy kényelmetlen a használatuk.
Nyílt szabványokat és protokollokat használ, ezek általában szöveg alapúak.	Lassabb, mint néhány elosztott rendszer. Ez a hátránya a szöveg alapú formátumnak.
A HTTP segítségével keresztüljutnak a tűzfalakon, azok megváltoztatása nélkül.	Túljut a tűzfalakon, melyeknek célja a blokkolás vagy ellenőrzés lenne.
Egyszerűen lehet kombinálni a különböző gyártóktól származó szoftvereket.	
Lehetővé teszik, hogy a szolgáltatásokat és komponenseket újrafelhasználjuk egy infrastruktúrán belül is.	

2.4.2. Az XML jelölő nyelv

A webszolgáltatások alapja az XML (Extensible Markup Language), mely egy kiterjeszthető jelölő nyelv az informatikában. Az XML az SGML (Standard Generalized Markup Language) egy egyszerűbb, szűkített változata, melyből a legelőnyösebb tulajdonságok maradtak meg. Mivel az SGML már több mint 10 éve létezik, ezért kiterjedt eszközrendszere van és a fejlesztők számára sem jelent újdonságot a nyelv megismerése.

Elsődleges célja, hogy jól strukturált, szöveges dokumentumok segítségével történjen az információcsere az interneten. Azért nevezzük kiterjeszthetőnek, mert nincs rögzítve a címkekészlete, mint mondjuk a HTML-nél, hanem tetszőleges új elemmel bővíthető. Az XML alkalmas az informatikában használatos adatstruktúrák ábrázolására. Előnye, hogy platform független, így szinte minden rendszerbe beépíthető.

Mivel az XML jelölő nyelv ismertetése, eszközrendszerének és szabványainak bemutatása önmagában is egy szakdolgozat témája lehetne, ezért nem venném sorra ezeket. Kizárólag a webszolgáltatásokhoz kapcsolódóan említem meg az XML –re épülő protokollokat. Később a 3. fejezetben a gyakorlati példához kapcsolódóan mutatom be a működéshez szükséges XML dokumentumokat.

2.4.3. XML-RPC és SOAP

Ahogyan már a webszolgáltatások bevezetésénél említettem, az XML alapú üzenetkezelés megvalósítását jelenleg vagy az XML-RPC –vel, vagy a SOAP segítségével tehetjük meg.

XML-RPC

Ez a protokoll az XML és a HTTP segítségével biztosítja a hálózaton át történő metódusok vagy függvényhívások létrehozását. Az XML-RPC egy egyszerű rendszer, amely tulajdonképpen távoli eljárás hívást jelent. Innen jön az RPC kifejezés, ami a Remote Procedure Call szavak rövidítése. Az XML-RPC a HTTP protokollt használja arra, hogy adatokat vigyen át a kliens gépről a szerver gépre. Az üzenetek leírására pedig az XML nyelvet használja fel. A kliensek megkeresik az XML-ben az eljárás

nevét és paramétereit, a szerver pedig hibát vagy választ ad rá. Az XML fájlok egyszerűek és ezért könnyen kezelhetőek.

Egy XML-RPC kérés a következőképpen néz ki:

```
<?xml version=”1.0” encoding=”utf-8”?>
<methodCall>
      <methodName>my.getValues</methodName>
      <params>
            <param><value>1000</value></param>
      </params>
</methodCall>
```

1. forráskód

Egy methodCall nevű elemet tartalmaz, mely megmondja a metódus nevét és paramétereit.

Egy XML-RPC válasz pedig így néz ki:

```
<?xml version=”1.0” encoding=”utf-8”?>
<methodResponse>
      <params>
            <param><int>5</int></param>
      </params>
</methodResponse>
```

2. forráskód

A válaszban egy methodResponse elem van, amely megadja a visszaadott értéket.

SOAP

Az XML-RPC –hez hasonlóan a SOAP (Simple Object Access Protocol) is platform független, és szintén XML –re épül. Mindkét protokollt számítógépek közötti információcserére használjuk, de mi a különbség a kettő között? Az egyik különbség, hogy míg az XML-RPC kizárólag HTTP-POST –on keresztül küldi az adatokat, addig a SOAP számos átviteli protokollt vehet igénybe. Hiába a sok lehetőség a SOAP-nál, középpontjában mégis a HTTP áll. A másik nagy különbség, hogy az XML-RPC egyszerűségéhez képest a SOAP jóval bonyolultabb, ezáltal több mindenre is képes. Bonyolultságával már bármilyen objektum továbbítására is alkalmas. Hátránya viszont, hogy mivel összetettebb, ezért lassabb is.

A SOAP használja az XML Névterek és XML Séma szabványokat. Minden elem névtereken található, amelynek használatával elkerülhetjük a névütközéseket, de nehezíti az olvasást.

Egy egyszerű SOAP kérés:

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://www.w3.org/soap-envelope/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd=" http://www.w3.org/2001/XMLSchema">
      <SOAP-ENV:Body>
            <ns1:getValues
            xmlns:ns1="urn:examples:myexample"
            SOAP-ENV:encodingStyle="http://www.w3.org/2001/09/
            soap-encoding/">
            <valuename xsi:type="xsd:string">1000</valuename>
            </ns1:getValues>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3. forráskód

Itt a kérés egy úgynevezett borítékban van megírva, ez a SOAP Envelope, amely szabályokat definiál a kommunikáció résztvevői közötti adatok egységbe zárásához. Tartalmaz egy fejlécet és egy törzs részt. A törzs részben hasonlóképpen az XML-RPC –hez megadjuk a metódust és a paramétereket. Ezenkívül a SOAP specifikációnak része az adattípusok kódolását meghatározó szabvány.

Egy SOAP válasz:

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://www.w3.org/soap-envelope/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd=" http://www.w3.org/2001/XMLSchema">
      <SOAP-ENV:Body>
            <ns1:getValuesResponce
            xmlns:ns1="urn:examples:myexample"
            SOAP-ENV:encodingStyle="http://www.w3.org/2001/09/
            soap-encoding/">
            <return xsi:type="xsd:int">5</return>
            </ns1:getValuesResponce>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4. forráskód

A válasz szerkezetileg megegyezik a kéréssel, viszont itt a visszatérési értéket adjuk meg, ami jelen esetben egy egész típusú érték lesz.

A SOAP protokoll implementációja mára szinte minden programozási nyelv része. Szakdolgozatomban felhasznált PHP script nyelv 2004 –ben megjelent 5-ös verziójába került bele először a SOAP. A legújabb verziója a SOAP –nak a 2007-es SOAP 1.2, mely már W3C ajánlás.

2.4.4. WSDL a szolgáltatás leírása

A WSDL (Web Services Description Language) a szolgáltatások leírásáért felel. Legújabb verziója a 2007-es WDSL 2.0, amely W3C ajánlás. A WSDL segítségével a kliens alkalmazás meg tudja találni a szolgáltatást és meg tudja hívni annak nyilvánosan hozzáférhető műveletét. Egy WSDL dokumentumban elemi szinten definiálni kell, hogy mit tartalmazzon a szolgáltatás. Ez a protokoll is az XML nyelvre épül és 4 különböző információt ír le:

Interfész információ: itt írjuk le a funkciókat
Adattípusra vonatkozó információ: minden kéréshez és válaszhoz
Összekapcsolhatósági információ: átviteli protokollok megadása
Címzési információ: a szolgáltatás helyének meghatározása

A WSDL hasonlóan a SOAP –hoz névtereket használ és a legegyszerűbb esetekben is terjedelmes méretű a forráskódja. A legtöbb programozási nyelv automatikus WSDL generálási lehetőséget biztosít a nyelvi osztályokhoz. A PHP nyelvhez is léteznek ilyen kiegészítők.

A PHP5 –ös verziója már tartalmazza a SOAP PHP osztályt, amely a WSDL leírás alapján elkészíti a SOAP XML –eket. Ezek segítségével történik a közvetlen adatkommunikáció.

Szakdolgozatom gyakorlati részéhez kapcsolódó PHP alapú webshopban én is használok SOAP szervert és hozzá tartozó WSDL fájlt, ezért példaként az 1. számú mellékletben helyeztem el az egyik WSDL fájlt.

2.4.5. UDDI a szolgáltatások felkutatása

Az UDDI (Universal Discovery, Description and Integration) webszolgáltatások felkutatására, regisztrálására és integrálására használt technikai specifikáció. A webszolgáltatások három alapvető elemének az egyike. A W3C által kiadott legújabb verzió az UDDI 3.0.

Az UDDI gyakorlatilag egy gyűjtemény, egy nyilvántartás, ahol a WSDL dokumentumokat tárolják. Úgy találták ki, hogy ha egy vállalat új szolgáltatást készít, azt feltölti az UDDI adatbázisába, akkor egy másik vállalat onnan kikeresheti és felhasználhatja. A feltöltésre és a keresésre az UDDI API-kat határoz meg.

Az UDDI –ben található adatokat három kategóriába lehet sorolni:

Fehér oldalak: Általános információkat tartalmaz a cégről. Tartalmazza a cég nevét, tevékenységét, valamint a címét is.

Sárga oldalak: Vagy a céget osztályozza, vagy a cég által ajánlott szolgáltatást.

Zöld oldalak: Technikai információkat tartalmaz az adott webszolgáltatásról. Van egy mutatója, amely a külső specifikációra mutat és van egy címe, amely a webszolgáltatás meghívásához szükséges.

Ahhoz, hogy két alkalmazás között működjön a kommunikáció nincs szükségünk UDDI –re. Ha kevés szolgáltatásunk van, akkor elhagyhatjuk, de ha több szolgáltatásunk van érdemes feltölteni UDDI gyűjteménybe.

Szakdolgozatomban szereplő gyakorlati feladatnál nem használok UDDI gyűjteményt.

2.5. Webáruházak szinkronizálásának módjai

Az előző fejezetekben már bemutattam, hogy mit is jelent a szinkronizálás és milyen technológiákat használhatunk fel az adatátvitel során. Ebben a fejezetben webáruházak szinkronizálásának módjait sorolom fel, minden módszerre élő példát hozva. A szinkronizálás módjait PHP környezetben vizsgálom, függetlenül attól, hogy B2C vagy B2B webáruházról beszélünk. Mivel ebből a témakörből még nem nagyon készült

szakirodalom a saját tapasztalataimat próbálom összegyűjteni a szinkronizálási módok elemzése során.

2.5.1. SOAP server segítségével

Ha webszolgáltatások segítségével történik az adatátvitel, akkor SOAP protokollt használva XML üzeneteket küldünk az egyik alkalmazástól a másikig. Abban az esetben alkalmazzuk ezt a módszert, ha teljesen automatizálni szeretnénk a folyamatokat, vagyis real-time kommunikációt szeretnénk létrehozni.

Importálás esetén a vállalat üzleti alkalmazása szolgáltatja a SOAP klienst és egy SOAP hívást kezdeményez. A hívás tartalmazza a szolgáltatás leírását, a webáruház SOAP szerverének WSDL elérési útját és azokat az adatokat, amiket importálunk. A WSDL fájl alapján elindul a SOAP szerver és a szolgáltatás lefut, amely feldolgozza az adatokat. Természetesen a külső rendszernek, mely legyen akár vállalatirányítási vagy ügyviteli rendszer, ismernie kell a webáruház leírását arról, hogy hogyan és milyen adatokat tud feldolgozni. Ez a leírás általában egy API (Application Programming Interface) dokumentációban található. Ilyenkor egy irányban, valós idejű szinkronizációt tudunk létrehozni. Például, ha azt szeretnénk, hogy a vállalat ügyviteli rendszerében a termékek módosításakor azonnal értesüljön a webshop a változásokról.

Exportálás során hasonlóképpen működik a folyamat csak fordítva. A webshop biztosítja a SOAP klienst, az üzleti alkalmazás SOAP szervere pedig fogadja az üzenetet. Ebben az esetben is ismernie kell a SOAP klienssel rendelkező félnek a másik fél szolgáltatásainak és adatainak leírását. Ilyenkor is létrehozható valós idejű kommunikáció, de mivel a cégek általában a rendeléseket naponta csak néhány alkalommal dolgozzák fel és a feliratkozókra is csak akkor van szükségük, amikor egy hírlevelet kiküldenek, ezért ebben az esetben az aszinkron megoldásokat szokták választani.

Ha egy általános megoldást szeretnénk a webáruházunkhoz készíteni, amely egy SOAP üzeneten keresztül feldolgozza a kapott adatokat, akkor ez egyszerűen megvalósítható. Egy ilyen megoldás esetén, ha biztosítunk egy API-t arról, hogy a webáruház hogyan dolgozza fel az adatokat és azoknak hogyan kell kinéznie, akkor bármilyen külső rendszerrel könnyen létrejöhet a szinkronizáció az importálás irányában.

Ellenkező esetben, viszont már nehéz egy olyan általános megoldást készíteni, hogy a webáruházunkból exportált adatokat, bármilyen külső rendszer SOAP szervere fogadni tudja, mivel ahány különböző külső rendszer van annyi API-t kellene ismernünk.

Speciális esetben viszont van arra lehetőség, hogy valós idejű kommunikációt hozzunk létre az adatátvitel mindkét irányában. Ekkor mindkét félnek szükséges fejlesztéseket végrehajtania a kommunikáció létrehozásának érdekében és API-kat biztosítania.

Egy ilyen real-time szinkronizációt ismerhettem meg szakmai gyakorlatom során az akkumulator.ws webáruháznál. Ennél a boltnál a megrendelők olyan szinkronizációt kértek, ami mindkét irányban valós időben zajlik. Ez úgy lett megvalósítva, hogy mind a webáruház, mind pedig a vállalatirányítási rendszer SOAP kliensként és szerverként is funkcionál. A webáruházban, amint lekéri egy felhasználó a termék adatait, vagy a kategória oldalon több termék adatait, a webáruház SOAP kliense értesíti a másik rendszer SOAP szerverét. Ezután az ERP lekéri a megfelelő adatokat és továbbítja a webáruház SOAP szerverének. Látható, hogy ez a folyamat teljesen automatikusan működik.

2.5.2. XML segítségével

Az XML jelölő nyelvet a 2.4.2. –es fejezetben mutattam be. Az XML segítségével jól struktúrált dokumentumot hozhatunk létre, melyben akár a termékek adatait is tárolhatjuk. A 2. számú mellékletben egy minta XML található, két teszt termék adataival. A szinkronizálás során ehhez hasonló XML fájlok állnak az adatátvitel középpontjában.

Gyakorlatban a webáruház rendszerhez készíteni kell egy komponenst, amiben az XML fájlból felolvassuk, illetve az adatbázisból egy XML fájlba mentjük az adatokat. Az XML felolvasáshoz és az XML generáláshoz komolyabb programozói tudás szükséges. Ismernünk kell az XML nyelv szerkezetét, szabályait, melynek segítségével az adatbázisból lekért adatokat összeilleszthetjük, és XML fájlba menthetjük. A felolvasáshoz szükségünk van egy XML értelmezőre, melyet a legtöbb programozási nyelv biztosít számunkra.

Az adatátvitel történhet manuálisan, ilyenkor a webáruház kezelője egy gombnyomásra letölti az XML fájlt, melyet az ügyviteli rendszerbe importál, vagy fordítva, amikor az

ügyviteli rendszer felületén kattint és elkészíti az XML fájlt, melyet a webáruházba importál.

Ahhoz, hogy az adatátvitelt automatizálni tudjuk, szükségünk van egy közös, mindkét fél által ismert könyvtárra, legyen ez vagy a webáruház, vagy pedig az ügyviteli rendszer szerverén. Az összeakadás elkerülése érdekében, amibe ír a webáruház, abból olvas az ügyviteli rendszer, illetve amibe ír az ügyviteli rendszer abból olvas a webáruház. A két mappa vagy az ügyviteli rendszer szerverén helyezkedik el, ahova a webáruház tölti fel a fájlokat FTP segítségével, vagy a webáruház szerverén található, ilyenkor az ügyviteli rendszer gondoskodik a feltöltésről.

Aszinkron átvitel esetén, időközönként, akár 10 percenként, vagy óránként automatikusan legenerálódik az XML fájl a webáruház adataiból a közös mappába, amit szintén meghatározott időközönként figyel az ügyviteli rendszer és onnan felolvassa az adatokat. Fordított esetben is hasonlóan működik.

Létezik egy másik automatizálási megoldás, amikor egy HTTP kérés GET vagy akár a POST paraméterének segítségével adjuk át az adatokat, melyek sztring típusúak és a teljes XML dokumentumot tartalmazzák. Mivel az URL cím hossza korlátozott figyelembe kell venni, hogy egy nagyméretű XML-t nem tudunk feldolgozni ezzel a módszerrel.

Ha teljesen valós idejű megoldást szeretnénk készíteni, akkor az előző fejezetben ismertetett SOAP protokoll segítségével történő kommunikációt érdemes használni.

2.5.3. CSV import, export

A CSV (Comma Separated Values) egy olyan fájl formátum, melyben vesszővel vannak elválasztva az adatok. A legtöbb szövegszerkesztő alkalmazás és a mai világban szinte már minden olyan szoftver, amely adatokat tárol, képes CSV formátumban exportálni és importálni.

A gyakorlati megvalósítása a CSV szinkronizációnak teljesen megegyezik az XML megoldáshoz, a különbség mindössze annyi, hogy itt más fájlformátumot használunk.

A CSV feldolgozáshoz és a CSV generáláshoz nem szükséges komolyabb programozói tudás. A különböző platformokon elég a fájl műveletek ismerése és a sztring kezelés,

ezek segítségével mindkét folyamatot el lehet végezni. Egyetlen dolog, melyet mindkét félnek ismernie kell, hogy milyen elválasztójelet használnak a szeparálásra. A szabványos megoldások a pontosvessző, vessző, de ettől eltérő megoldások is lehetségesek. Ha lehet ne bonyolítsuk meg a fejlesztők dolgát egy dollár ($) vagy egyéb speciális karakterekkel, mert abból csak probléma adódhat.

Talán, amiért a CSV formátumot érdemes használni az, hogy az ügyviteli rendszerünk, nem képes csakis kizárólag CSV fájlok kezelésére. Ilyen esetekben nagyobb munka a szoftvert tovább fejleszteni más formátumok kezelésére, mint a CSV szinkronizációt választani.

A folyamat itt is automatizálható, de teljesen valós idejű átvitel, ebben az esetben nem, vagy nagyon bonyolultan valósítható meg, ilyen esetben célszerűbb egy másfajta szinkronizálási módszert választani.

2.5.4. XLS import, export

Az XLS a Microsoft Office irodai szoftvercsomag Microsoft Excel táblázatkezelőjének formátuma. A legújabb verziótól már az XLSX formátumot használják. Mivel ez egy speciális formátum, a különböző platformokhoz szükség van egy XLS kiterjesztésre, mely értelmezi a táblázatkezelő formátumát.

Az XML és CSV szinkronizációhoz hasonlóan a folyamat itt is ugyanúgy történik. A webáruház felolvassa az XLS-t, amit a külső alkalmazás legenerál és fordítva. Érdemes azonban átgondolni, hogy van-e értelme az XLS formátumot választani.

Az ügyviteli rendszer adatait nehezebb XLS formátumba kimenteni, mint CSV fájlokba. Ha nagyon ragaszkodunk az XLS –hez, akkor mentsük le CSV-be az adatokat és a táblázatkezelő képes beolvasni és táblázattá alakítani. Így már könnyebb a CSV fájlokkal történő szinkronizáció. Ha mégis az XLS-t választjuk, akkor számolnunk kell az ügyviteli rendszer továbbfejlesztésének költségével. Még egy indok a CSV fájl mellett, hogy a táblázatkezelő nem csak adatok tárolására lett kitalálva, hanem adatokkal való számolásra, grafikonokon való megjelenítésére, stb. Ezért mivel több mindenre képes egy Excel dokumentum, nagyobb méretű, mint egy egyszerű CSV fájl. Több száz terméket tartalmazó adatbázis esetén gondoljunk bele, mennyire lelassíthatja a rendszert.

E módszer esetén nem érdemes valós idejű adatátvitelben gondolkozni, mert ha meg is valósítható, több problémát okoz egy ilyen rendszer felépítése, mintha egy másik formátumot választanánk.

2.5.5. Megosztott adatbázis

Egy speciális szinkronizálási módszer, amikor nem használunk fájlokat, hanem egy közös, megosztott adatbázison keresztül cserélődnek az adatok. Van a webáruháznak egy saját adatbázisa, van az ügyviteli rendszer adatbázisa és van egy megosztott adatbázis, amely valamelyik fél szerverén található. A megosztott adatbázis bizonyos tábláiba a webáruház ír, ezekből az ügyviteli rendszer olvas, és azokba a táblákba, melybe az ügyviteli rendszer ír, abból olvas a webshop.

Gyakorlatban ez a megoldás úgy működik, hogy a két fél, vagyis a webáruház és az ügyfél eldönti, hol szeretné tárolni a megosztott adatbázist. Ha például az ügyviteli rendszer szerverén hoznak létre egy közös adatbázist, akkor az elérhetőségeit, a hozzáférési portot, felhasználónevet és jelszót biztosítaniuk kell a webshop fejlesztőinek részére. A rendszer elkészítése során le kell specifikálni, hogy milyen adatbázistáblákat használnak írásra a webáruháznál, melyiket olvasásra és persze milyen adatok kerüljenek a megfelelő táblákba. Ugyanezt a külső rendszernél is meg kell határozni.

Látszik tehát, hogy a megosztott adatbázissal történő szinkronizáció esetén mindkét félnél fejlesztésekre van szükség. Ezzel a megoldással az IT-Szalon webáruház fejlesztésekor találkoztam.

2.5.6. Kézi feltöltés

Ezt a módszert csak megemlítésként vettem bele dolgozatomba. Mivel a szinkronizáció lehet manuális is, vagyis ilyenkor emberi beavatkozás során történik az adatcsere, ezért beszélhetünk, erről a módszerről. Kézi feltöltés alatt, azt értem, amikor az adminisztrátor az ügyviteli rendszerében szereplő adatokat, kézzel egy adminisztrációs felületen keresztül, űrlapok segítségével feltölti.

Akkor használják ezt a módszert, amikor kevés terméket árul valaki és ezek a termékek néhány paramétert tartalmaznak. Tulajdonképpen ezt termékfeltöltésnek nevezzük, melyet azért veszek a szinkronizálási módszerekhez, mert a szinkronizáció

meghatározásából kiindulva, termékfeltöltés során is azt az eredményt kapjuk, hogy mindkét rendszerben megegyeznek az adatok.

Egy webáruház és egy ügyviteli rendszer alapból úgy van elkészítve, hogy az adatokat kézzel fel tudják tölteni. Azok számára, akiknek az előző fejezetekben leírt szinkronizáláshoz kapcsolódó információk mégsem tetszenének és saját maguk szeretnék kezelni az adataikat, ezt a módszert ajánlom.

2.6. Szinkronizálási módok összehasonlítása

Elérkeztünk a szakdolgozatom témájának választott szinkronizálási módok összehasonlításához. A módszerek bemutatása után különböző szempontok szerint hasonlítom össze az előzőekben ismertetett szinkronizálási lehetőségeket. A szempontokat fontossági sorrend szerint veszem figyelembe, melyek a költség, a gyorsaság, megbízhatóság és biztonság.

2.6.1. Költség szempontjából

Talán a legfontosabb szempont egy vállalat életében a költségek meghatározása. Függetlenül attól, hogy egy nagyvállalatról vagy egy kisebb cégről beszélünk, a szinkronizálási mód kiválasztását a pénz határozza meg. A webáruház elkészítése során fel kell mérnünk az igényeket, milyen módszer áll legközelebb a vállalat elképzeléseihez.

Amennyiben egy komoly infrastruktúrával rendelkező vállalatirányítási rendszer áll a cég rendelkezésére, valószínűtlennek tartom, hogy ne egy valós idejű kommunikációt választanának, mely webszolgáltatásokat használ fel. Természetesen ennek a költségei is magasabbak, de a valós idejű kommunikáció megvalósítása ezzel jár. Vannak azonban olyan üzleti csomagok, melyeket kompletten meg lehet úgy vásárolni, melyben minden szolgáltatás és szoftver is megtalálható. Az ilyen rendszerek általában beépített webszolgáltatásokat is tartalmaznak, melyeket felhasználhat a vállalat külső kommunikációra. Egy ilyen alkalmazáscsomagot, viszont meg is kell fizetni, amire nem minden cégnek van lehetősége.

Ha a vállalat úgy gondolja, hogy már nem fér bele a költségvetésébe a valós idejű kommunikáció és gyakorlatilag nem számít nekik, hogy az adatokat 10 percenként vagy

óránként frissítik, akkor egy aszinkron módszer használata is elegendő. Ebben az esetben az adatátvitel eszköze lehet akár egy XML, CSV vagy XLS fájl is.

A költség szempontjából azonban figyelembe kell venni, hogy egy kisebb vállalkozás számára mi az elégséges szinkronizálási módszer és mi egy nagyvállalat esetén. Mivel egy nagyvállalatnál naponta érkezhetnek új termékek a raktárba, ezért ott elengedhetetlen, hogy ne egy aszinkron megoldást, hanem egy valós idejűt válasszanak.

2.6.2. Gyorsaság szempontjából

A gyorsaság a második legfontosabb szempont az informatikan és a webáruházak világában is. Mivel értékesítéssel foglalkozunk, melyből a bevételünk származik, oda kell figyelnünk a sebességre. Ha a webáruházunk lassú működése elriasztja a vásárlókat, bevételtől esünk el, amit egyik vállalat sem szeretne. Úgy kell megválasztanunk a szinkronizálási módot, hogy minden lehetőség figyelembe vételével, egy átlagos sebességgel működő rendszert hozzunk létre.

Olyan szinkronizálási mód esetén, amikor valós időben történik a kommunikáció és automatikusan történik az adatátvitel, figyelnünk kell arra, hogy milyen sávszélességgel tud kommunikálni a webáruház és milyennel a vállalat ügyviteli rendszere. Ha az egyik szerver lassabb sebességet produkál, akkor a másik rendszert is belassítja. Ha már a vállalat ezt a megoldást választja és kifizeti a fejlesztéseket, ügyeljen, arra is hogy a megfelelő sávszélességet is biztosítsa a rendszer optimális működéséhez.

Aszinkron adatátvitel esetén is fontos lehet a gyorsaság. Amikor az adatokat frissítjük (legyen az 10 percenként, vagy havonta) és a fájlok, amikben az adatokat tároljuk több megabájt méretűek, akár hosszú perceket is várnunk kell a feltöltésre. Ilyenkor benne van a veszély, hogy a hosszas futás alatt megáll a feltöltés, vagy esetleg a memória betelik és kezdhetjük az elejéről. Ha egy hiba miatt leáll a feltöltés, a vásárló, aki éppen a webáruház termékeit böngészi, hibás adatokat láthat.

Megosztott adatbázis használata esetén a közös adatbázis elérésének a sebességét kell vizsgálnunk. Amennyiben lassan tudja elérni a webáruház az adatbázist, akkor nem biztos, hogy ez a megoldás a legcélravezetőbb. Ha a webáruház szerverén található a közös adatbázis, akkor azt gyorsan el tudjuk érni, viszont ilyenkor a külső rendszer szempontjából kell vizsgálni az elérhetőséget.

2.6.3. Megbízhatóság szempontjából

A megbízhatóság szükséges ahhoz, hogy egy jól működő webáruházat üzemeltessünk. Hiába van felkészítve a webshop és az ügyviteli rendszer a szinkronizációra, ha működése nem megbízható. Megbízhatóság alatt itt azt értem, hogy egy szinkronizáció mennyire használható, hibamentes és karbantarható.

Ebből a szempontból nézve a leghasználhatóbb és a legtöbbet is használt módszerek az aszinkron adatátvitelt megcélzó módszerek. Mivel egyszerű a működésük, egy informatikában kevésbé jártas ember is megértheti őket. Egy bonyolultabb megoldás, ami megosztott adatbázist vagy SOAP protokollt használ, már nehezebben fogható fel.

A hibamentesség alapján mindegyik megoldásról elmondható, hogy előfordulhatnak hibák, azonban úgy gondolom, hogy a beüzemelés után, a tesztelési folyamat során a legtöbb hibát ki kell küszöbölni. Később is előkerülhetnek hibák, amelyek csak speciális esetekben fordulnak elő. Sokkal nagyobb problémát okozhat egy SOAP szerver vagy megosztott adatbázis esetén egy hiba.

A karbantartás is a megfelelő működéshez szükséges. Ezt a feladatot azok végzik, akik elkészítették a két rendszer közötti szinkronizációt. Gyakran a webáruház fejlesztők felelnek az ő részükért, a külső rendszer karbantartói pedig a sajátjukért. Saját meglátásom szerint bármilyen módszert választunk, a karbantartás könnyen végrehajtható minden esetben, mivel mindenki ismeri a saját munkáját.

2.6.4. Biztonság szempontjából

Azok az átviteli módok, amik során az üzenet a HTTP protokollon keresztül megy át nem túl biztonságosak, mivel a tűzfalak is engedik a HTTP-n való átvitelt. Ezért ezekben az esetekben szükségünk van biztonsági megfontolásokra. Webszolgáltatások esetében a biztonság kérdése központi téma. Három fontos területe van: bizalmas információk kezelése, hitelesítés és a hálózati biztonság. Mivel sok féle megoldás létezik ezek megvalósítására, biztosak lehetünk abban, hogy amennyiben használjuk is ezeket a biztonsági protokollokat, a SOAP protokoll segítségével történő adatátviteli mód biztonságos.

A HTTP protokollon keresztül történő egyéb szinkronizálási módok esetében is léteznek biztonsági megoldások. Ha még biztonságosabb módot szeretnénk használni, lehetőség van a HTTPS protokoll használatára is.

Apache webszerverek esetében, a htaccess és htpasswd fájlok segítségével lehet egy URL-t jelszóval levédeni, amivel biztonságosabbá tehetjük rendszerünket.

FTP protokoll használata esetén és a megosztott adatbázisoknál is fontos hogy a kapcsolódási adatokat titkosítsuk, úgy hogy ne legyen hozzáférhető más külső fél számára.

Biztonsági szempontból tehát elmondható, hogy minden módszernél szükség van arra, hogy megfelelően védjük a kommunikációt és adatainkat. Mindegyik esethez különféle biztonsági módszerek alkalmazhatóak, amelyeket összehasonlítva talán a webszolgáltatások védelme tűnik a legbonyolultabb feladatnak.

3. Egy szinkronizálási módszer bemutatása

Szakdolgozatom második nagy részében egy olyan rendszert mutatok be, mely egy általános megoldást nyújt webáruházaknak külső rendszerekkel való szinkronizációjára. Ezt a szinkronizációt SOAP protokoll segítségével valósítottam meg.

Az elméleti részben már írtam arról, hogy a valós idejű kommunikáció megvalósítására nem lehet általános megoldást létrehozni úgy, hogy mindkét irányban, bármilyen külső rendszerrel működjön az automatikus szinkronizáció. Gyakorlati példámban úgy készítettem el az alkalmazást, hogy az egyik irányban képes legyen automatikus kommunikációra.

A szinkronizációs megoldást PHP környezetben hoztam létre és egy SOAP szervert használtam az adatátvitelhez. Ebben a fejezetben ismertetem, hogy miért hasznos egy ilyen megoldás, hogyan működnek a folyamatok és ezeket kódszinten is bemutatom.

3.1. Probléma ismertetése

Adott egy webáruház motor, amelyhez egy általános megoldást szeretnénk készíteni, arra az esetre, ha egy külső rendszerrel szeretnénk szinkronizációt létrehozni. Ahhoz,

hogy ne a webáruház fejlesztőknek kelljen a különböző ügyviteli rendszerekhez alkalmazkodni egy általános API-t kell készíteni, melyben leírjuk, hogy a webáruházunk szinkronizációs komponense, hogyan és milyen adatokat képes feldolgozni. Nagy előny származhat egy ilyen komponens és API létrehozásából, ugyanis így nincs szükség minden webáruház esetén egyedi, az ügyviteli rendszertől függő fejlesztésekre. Ezenkívül mivel egy általános megoldásról beszélünk, a szinkronizációs rendszer bekonfigurálása is kevesebb időt vesz igénybe mindkét fél számára.

3.2. Fejlesztői környezetem

A webshophoz egy Joomla! 1.0.11 Stable CMS (Content Management System) tartalomkezelő rendszerbe épített VirtueMart 1.0.7 Stable webáruház komponenst használtam fel. A Joomla CMS-t és a VirtueMart komponenst letölthetjük a http://www.joomla.org.hu, illetve a http://www.virtuemart.net oldalakról.
A szinkronizációhoz számunkra szükségtelen komponenseket és modulokat kikapcsoltam a tartalomkezelő rendszerben, annak érdekében, hogy jobban szemléltetni tudjam a szinkronizációs komponens működését.

A webáruház rendszert egy XAMPP fejlesztői környezetben fejlesztettem, ami egy ingyenes programcsomag. Az XAMPP 1.7.1 es verziószámú csomagját használtam, amely tartalmaz egy Apache 2.2.11 –es web szervert, egy MySQL 5.1.33 –as adatbázis szervert, PHP 5.2.9 –es PHP fordítót és egy phpMyAdmin 3.1.3.1 –es adatbáziskezelő felületet. Ezzel a programcsomaggal a saját számítógépemen hozhattam létre egy éles webszervert, amely képes adatbázis kezelésre és PHP futtatásra, ezek elengedhetetlen kellékei a fejlesztésnek.

Az XAMPP fejlesztői környezetet le tudjuk tölteni a hivatalos oldalukról a http://www.apachefriends.org/en/xampp.html címen.

3.3. A rendszer bemutatása gyakorlati szemmel

A rendszer működését a folyamat állomásainak sorrendjében fogom ismertetni. Első lépésben a SOAP kliens működését írom le, majd kitérek a WSDL fájlokra. Ezután a SOAP szerver kerül bemutatásra. A szerver bemutatása során az `XmlService`

szolgáltatást és a `WseInput()` metódusát értelmezem. A további részekben a naplózási folyamatot, a validációs réteg műveleteit, az objektum transzformálását, majd a perzisztencia réteget ismertetem. A fejezet végen arra az esetre, ha nem szeretnénk SOAP –ot használni, bemutatok egy megoldást, amely felparaméterezett URL-t használ, végül a rendelések exportálását is leírom.

3.3.1. A SOAP kliens működése

A külső rendszert a **wsesync/soapcall.php** fájl fogja reprezentálni, amely tartalmaz egy objektumot, a SOAP klienst és a függvényhívást. A külső rendszerbe ehhez hasonló kódrészt kell beépíteni. Egy termék esetében, amikor a külső rendszerben módosítás történik, akkor a termék lesz az objektum, melyet a SOAP kliens a függvényhíváson keresztül továbbküld. A PHP nyelv az 5 –ös verziójától már rendelkezik SOAP osztállyokkal. A SOAP klienst a `SoapClient` osztály példányosításával hozzuk létre, ami paraméterként a WSDL szolgáltatás leíró fájl elérési útját várja. Miután létrejön a SOAP kliens, a WSDL fájlban leírt függvényhívás hajtódik végre. A WseInput függvénynek átadjuk az objektumot, mely a termék adatait tartalmazza. Mivel hibával is visszatérhet a függvény ezért hibakezelés szükséges.

3.3.2. A WSDL leíró fájl

Az 1. számú mellékletben látható a WSDL leíró fájl (**wsesoapserver01.wsdl**), melynek segítségével, leírjuk a szolgáltatás típusait, függvényeit és a szükséges adatokat, mint például a SOAP szerver elérési útját. A mellékletben látható, hogy két műveletünk van: a WseInput a bejövő üzenetet dolgozza fel, a WseInputResponse a választ adja az üzenetre.

A PHP nyelv nem egy típusos nyelv. Az objektumnak, amit átadunk nincs típusa, a nyelv sztringként értelmezi. Ha a külső rendszer PHP nyelven lett fejlesztve, akkor nem okozhat problémát az objektum átadása, azonban egy .NET vagy egy JAVA alapú rendszerben lévő objektumokkal nem tud mit kezdeni a PHP. Szükséges tehát egy olyan WSDL leíró is (**wsesoapserver02.wsdl**), ahol a beérkezett adatok egy egységes, közös formában vannak megadva. Erre nagyon jó megoldást nyújt egy XML dokumentum. A 2. számú WSDL fájl a 2. számú SOAP szervert fogja elindítani. Ha összehasonlítjuk a két WSDL fájlt nem sok különbséget találunk. Az eltérés mindössze annyi, hogy a 2.

számú SOAP szerver a beérkezett XML fájlból PHP objektumot hoz létre. A 2. számú SOAP szerver és WSDL fájl a CD mellékleten található.

3.3.3. SOAP szerver tartalma

A SOAP szerver indítására is létezik osztály a PHP-ben. A `SoapServer` osztály konstruktora szintén a WSDL leíró elérési útját várja paraméterként. A CD mellékleten megtalálható a **wsesoapserver01.php** és **wsesoapserver02.php**, a különbséget már az előző pont végén említettem. Ezekben a fájlokban írjuk meg a szolgáltatást, mely a WSDL fájl szerint XmlService nevet kell viselnie. Az **`XmlService`** nevű class egyetlen WseInput nevű metódust tartalmaz, ami a WSDL fájlnak megfelelő.

A függvény legelején megvizsgáljuk, hogy létezik-e az objektumunknak wsesync nevű eleme, ha nem akkor ezt hibaüzenetként jelezzük. A wsesync nevű elem jelzi számunkra, hogy a küldött objektum termék, kategória, raktárkészlet, rendelés állapot vagy termék láthatóság lesz. A következő részben a beérkező objektumot egy txt fájlba írjuk, ami a data/temp mappában jön létre, ezzel nyoma marad a kérésnek. Ezután a **`WSESync`** osztály **`handleTheObject()`** nevű statikus metódusát hívjuk meg. Ennek a függvénynek egyik paramétere maga az objektum, a másik paramétere pedig az objektum wsesync nevű eleme, ami a típust határozza meg.

3.3.4. Naplózás a műveletek előtt

A szinkronizációhoz megírt osztályokat, vagyis az XML feldolgozó, validációt végző, adatokat letároló, naplózó osztályokat, a classes mappában helyeztem el. Ezek az osztályok automatikusan töltődnek be a **classes/loader.php** segítségével. A WSESync nevű mappában található a **wsesync.class.php** a már említett **`handleTheObject()`** statikus metódussal. A naplózás a **`wsesynclog`** osztály segítségével történik. Egyetlen metódusa van, a statikus **`write()`** metódus, mely egy adatbázis táblában letárolja a számunkra később fontos adatokat. Nézzük mik lesznek ezek az adatok:

- time: az időpont amikor a kérés beérkezett (timestamp)
- backtrace: a visszakövetés miatt tároljuk le, ez egy sztring lesz, ami az aktuális fájl elérési útját és az aktuális függvényhívást tartalmazza
- location: paraméterként kapja meg a függvény, ez jelen esetben a HandleTheObject – start sztring lesz

- name: az bejövő objektum típusa (termék, kategória, stb.)
- response: érkezett-e válasz a kérésre, mivel még a kérést nem dolgoztuk fel, ezért 0.
- response_code: a válasz hibakódja, amikor még nem érkezett rá válasz, akkor -1, ez az alapértelmezett, amikor hiba nélkül végrehajtódott a kérés, akkor 0, amennyiben hibásan ért véget 1.
- data: a beérkezett objektumot szérializáltan tároljuk le.

Az adatbázisban való letároláshoz a DBO adatbázis kezelő osztályt használom. Ennek az osztálynak a segítségével, mind a tárolási, módosítási, törlési és lekérési műveleteket el tudom végezni.

A write metódus az 1 hónapnál régebbi napló bejegyzéseket törli, mivel gyakori kérések esetén hamar megtelhet az adatbázis.

3.3.5. A bejövő objektum nevének validációja

Folytatjuk a `handleTheObject()` metódus elemzését, aminek következő programrészében az **administrator/components/com_wsesync/wsesync.config.php** fájlból felvesszük a MAIN_XML konstanst. Ez tartalmazza a validációs XML elérési útját, jelen esetben a következő útvonalat:

administrator/components/com_wsesync/IO/validation/validation.xml

A validációhoz a **`ValidationHelper`** osztályt hívjuk segítségül.

Konkrétan a `ValidationHelper` osztály **`getValidationXMLName()`** metódusa végzi a validációt, melynek első paramétere az input objektumot azonosító név, második paramétere pedig a `xmlMainObj` nevű objektum, ami a validation.xml –ből létrehozott objektum (`XmlObject` osztály segítségével).

A `getValidationXMLName()` metódus mindössze annyit vizsgál meg, hogy a bejövő objektum neve létező név-e a validation.xml –ben, és ha igen akkor visszaadja az objektumhoz tartozó transzformációs XML fájl nevét (`validationFile`). Ez azt jelenti, hogy ha a bejövő objektumot azonosító név a product, akkor mivel van ilyen nevű elem a validation.xml-ben, ezért visszakapjuk a **product.xml** –t, ami a termékekhez tartozó transzformációs xml fájl.

Természetesen mind az 5 objektum azonosítóhoz tartozik transzformációs fájl, ezek a fájlok szintén az **administrator/components/com_wsesync/IO/validation/** mappában találhatóak.

3.3.6. Transzformálás és szükséges adatok vizsgálata

Miután megtaláltuk a transzformációs xml fájlt, készítünk egy objektumot az `XmlObject` osztály segítségével belőle. Ez lesz a `validationObj` objektum.

A transzformált objektumot a `validatorHelper` osztály **`transformAndCheckRequired()`** statikus metódusa készíti el a bejövő objektumból és a `validationObj` objektumból. Egy példán keresztül szeretném szemléltetni a transzformálást. Kategória esetén a bejövő objektumunk a következő adatokat tartalmazza ID = 5, NAME = Kategória 5, PARENT = 0. A transzformálás során egy olyan objektum jön létre, aminek elemei a következők lesznek: category_id = 5, category_name = Kategória 5, parent_id = 0. A transzformált objektum elemeinek neve megegyezik az adatbázistáblában szereplő mező nevekkel.

Egy fontos feladata még van ezenkívül a `transformAndCheckRequired()` metódusnak, méghozzá a szükséges adatok vizsgálata. Ha a transzformációs XML elemei rendelkeznek required attribútummal, akkor azokat is vizsgálja. Amennyiben a required attribútum értéke true, és üres a csomópont, akkor hibával tér vissza a metódus.

3.3.7. Az adatok szemantikai elemzése

A szemantikai elemzés alatt azt értem, hogy a bejövő adatok olyan formában legyenek, ahogyan a webáruház rendszer megkívánja, gyakorlatilag ez egy logikai elemzés. A kategória példánál maradva, egy category_id –nak és parent_id -nak számnak, a category_name nek pedig szövegnek kell lennie. Azonkívül, hogy a category_name szöveg, még azt a feltételt is megadtam, hogy nem lehet kisebb 3 karakternél a hossza.

A `validatorHelper` osztály **`getValidationClassName()`** metódusa olvassa ki a validation.xml fájlból a validationClass nevét. A példánk esetében a kategóriákhoz a categoryValidator név tartozik. Amennyiben meg van adva az xml-ben a validationClass neve és meg van írva ez az osztály (**categoryvalidator.class.php**), akkor a loader.php be fogja tölteni. Azonban, ha nincs megírva az osztály létre kell

hoznunk egyet, ami gyakorlatilag nem csinál semmit. Ezt a `ValidatorFactory` osztály teszi meg nekünk.

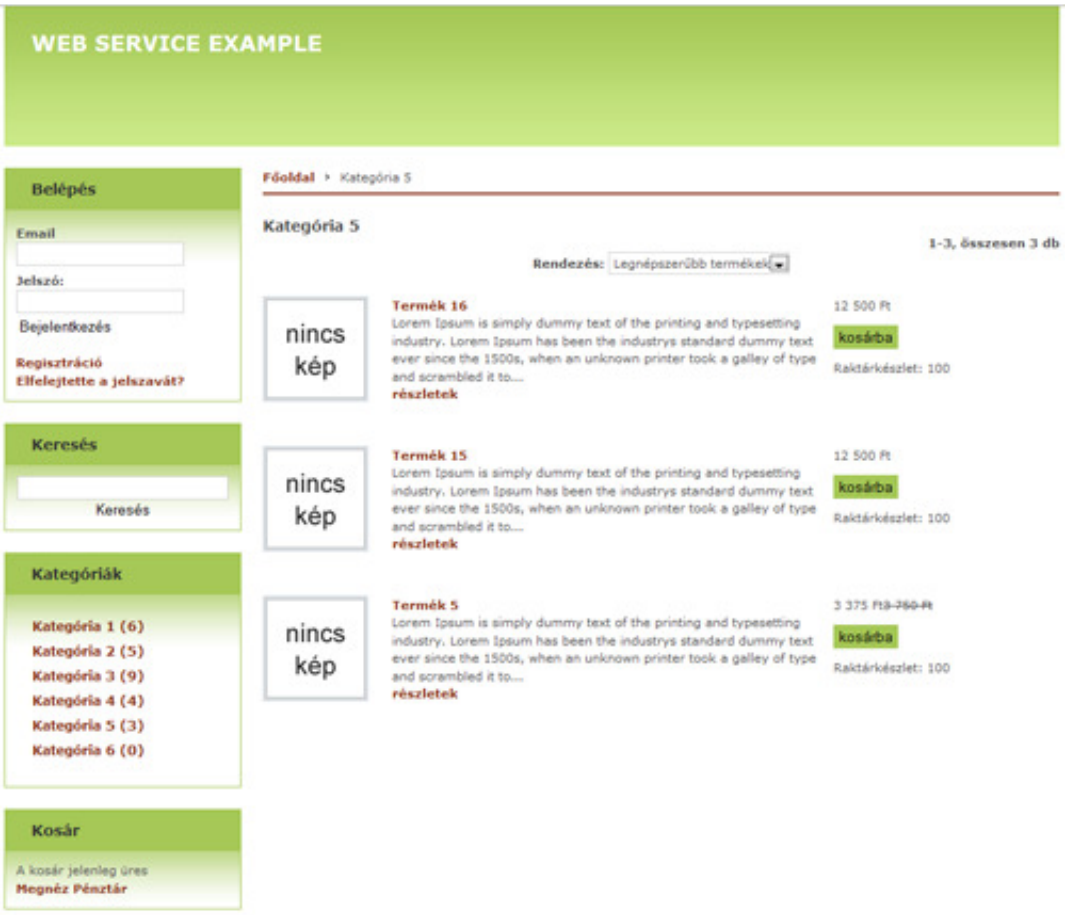
3.3.8. A perzisztencia réteg

Miután naplóztuk, validáltuk és transzformáltuk a bejövő adatokat, ezután már csak annyi hiányzik, hogy a megfelelő módon le tároljuk őket. Az objektumok letárolására plugin osztályokat hoztam létre, amelyek elvégzik a feladatokat. Ezeket a pluginokat is a validation.xml –ben adom meg és a ValidatorHelper osztály **`getPluginName`**() metódusával veszem fel. A pluginokat a **`WseSyncPlugin`** osztály fogja össze, melynek van egy **`getOne()`** metódusa, ennek segítségével a példányosított plugin osztályt kapjuk meg, és van egy **`handle()`** absztrakt metódusa, amit minden plugin örököl, és az egyes plugin osztályokban felüldefiniálunk.

A kategória példánkat megnézve a plugin neve Category a validation.xml –ben, ami alapján a `getOne()` metódus példányosítja a **`CategoryPlugin`** osztályt. Ez az osztály pedig a **classes/wsesync/plugin/categoryplugin.class.php** fájlban van megírva. Ezek a plugin osztályok gyakorlatilag a megkapott adatok alapján, a megfelelő adatbázis táblába letárolják az adatokat. Ezt a műveletet a `handle()` metódus végzi. Amennyiben létező azonosítót adunk meg, akkor módosítja az adott kategóriát, terméket vagy a többi objektum típust. Néhány egyedtől függő vizsgálatot is végzünk a `handle()` metódus elején, például kategória esetén, ha szülő kategóriát megadunk, létezik-e olyan kategória, vagy rendelés esetén létezik-e a megadott rendelési állapot.

A perzisztencia rétegben lefutó műveletek eredményét, vagyis a visszatérési értékeket, amik lehetnek hibaüzenetek is, egy **persist_response** nevű változóban gyűjtjük össze. Ehhez a PHP ob_start függvényét használjuk, amivel a kimenetre való pufferelést tudjuk bekapcsolni. Amíg le nem zárjuk a pufferelést, minden a kimenetre küldött adat egy belső pufferban tárolódik el.

Miután az adatokat a plugin segítségével letároltuk, a webáruház frontend felületén már láthatóak is a változások. A következő screenshoton már látható, hogy a termékkategóriánk Kategória 5 néven szerepel.



2. ábra *Kategória 5 megjelenése a frontenden*

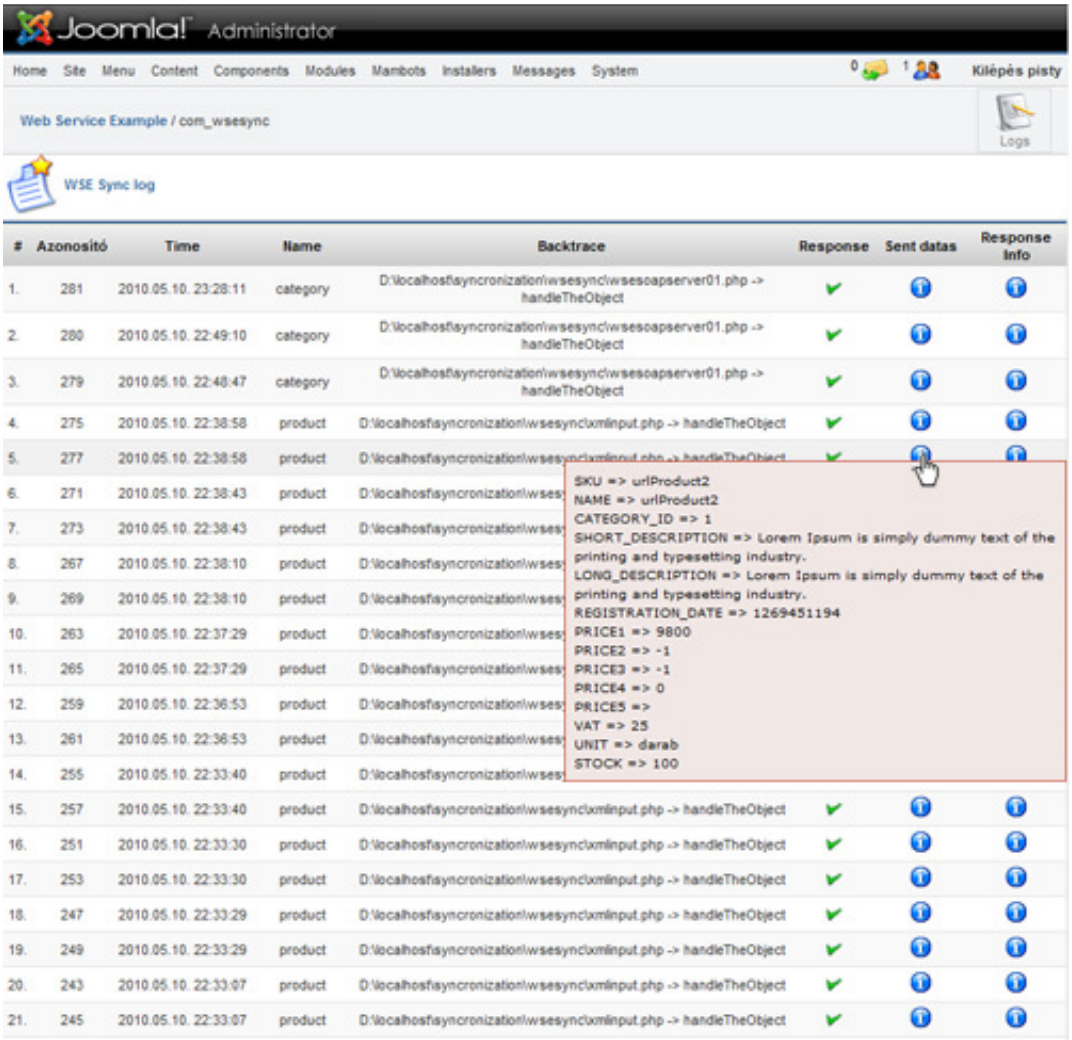
3.3.9. Naplózás a műveletek után

Az adatok letárolása után újra szükséges, hogy naplózzunk. Az előző rész végén említett `persist_response` változó tartalmát átadjuk a WseSync osztály `response()` metódusának, ami beállítja a statikus response_text adattag értékét. Ez akkor történik meg, ha létezik a persist_repsonse változó. Ezután újra a wsesynclog osztályt hívjuk segítségül a `write()` metódusával és hasonlóan a 3.3.4. –es fejezetben leírtakhoz, itt is letároljuk az adatokat. Annyi a különbség, hogy a response mezőbe annak a kérésnek a napló táblabeli azonosítóját tároljuk le, amire ez a válasz érkezett. Ezenkívül a location mező is értelemszerűen a következő lesz: handleTheObject – end.

A naplózás megjelenítésére az adminisztrációs felületen biztosítottam lehetőséget. Az adminisztrációs felületen, a következő url segítségével érhetjük el a naplóbejegyzések listáját:

administrator/index2.php?option=com_wsesync

A 3.3.4. es fejezetben ismertetett mezőket és a választ ezen az oldalon lehet megtekinteni. A következő ábrán látható, a naplóbejegyzések listája.



3. ábra *Naplóbejegyzések listája az adminisztrációs felületen*

3.3.10. Befejezés, válaszadás

Miután a választ naplóztuk visszaadja a működést a `handleTheObject()` metódus a SOAP szervernek. Az `XmlService WseInput()` metódusa a WseSync osztály két statikus adattagját adja vissza, amik a response_code és a response_text tagok.

A szolgáltatásunkat, vagyis az `XmlService` osztályt, átadjuk értelmezésre a 3.3.1 –es fejezetben leírtak szerint példányosított SoapServer –nek. A PHP beépített SOAP

osztályának **`setClass`**() metódusának adjuk át a szolgáltatásunk nevét. Ezután meghívjuk a SOAP szerver **`handle`**() metódusát, ami szintén beépített függvény a PHP nyelvben.

3.3.11. Szinkronizáció felparaméterezett URL segítségével

Az azonnali kommunikáció megvalósítására a SOAP protokoll használatán kívül készült egy másik megoldás is. Ebben az esetben egy felparaméterezett URL segítségével kapjuk meg az adatot. Két kötelező paramétert kell megadni, az egyik az, hogy milyen típusú adat érkezik, amit a type paraméterrel adunk meg, a másik pedig egy xml fájl, amiben az értékek BASE64 kódoltak. A feldolgozást a **wsesync/xmlinput.php** fájl végzi.

Első lépésben a beérkező xml fájlt a **wsesync/xml** mappában tároljuk le. Ezután az `XmlObject` osztály segítségével objektumot készítünk belőle. Mivel az XML –ben itt egyszerre több terméket, vagy kategóriát is megadhatunk, ezért egy ciklusban járjuk be az összeset. A ciklus magjában a már jól ismert **`handleTheObject()`** metódus hajtódik végre. Miután letároltuk az adatokat és visszatért a `handleTheObject()` metódustól a működés, egy válasz üzenetet készítünk XML formában, amit a kimenetre írunk.

Egy felparaméterezett URL segítségével történő adatátvitelre hoztam példaként a 4. számú mellékletben található kódot. Ennek segítségével 2 termék kerül feltöltésre/frissítésre. A visszakapott válasz a következő ábrán látható:

```
−<responses>
  −<response>
      <code>0</code>
      <text_0>A urlProduct1 termék lementve!<br/></text_0>
    </response>
  −<response>
      <code>0</code>
      <text_0>A urlProduct2 termék lementve!<br/></text_0>
    </response>
  </responses>
```

4. ábra *Felparaméterezett URL hívásra kapott válasz*

Ha nem szeretnénk SOAP szervert és klienst használni, akkor a külső rendszerbe úgy építhetjük be ezt a módszert, az automatikusság megtartása mellett, hogy termék

feltöltés vagy módosítás esetén az alkalmazás automatikusan generálja le az XML fájlt és hívja meg az xmlinput.php-t a korábbiakban leírtak szerint.

3.4. Példa az exportálási folyamatra

A korábbi fejezetekben említettem, hogy egy általános megoldást nem tudunk készíteni az exportálás irányába. Arra viszont van megoldás, hogy például egy vagy több rendelést, XML dokumentumba mentsünk le. Ez azért nem nyújt általános megoldást, mert a külső rendszereknél nem biztos, hogy egy rendelést ugyanúgy reprezentálnak, mint a saját webáruházunk esetében.

Az exportálást a **wsesync/wsesync_order_export.php** fájl végzi, melyben a lényeget a következő sor végzi:

```
WSESyncExport::order(array(86,87),'');
```

A **`WseSyncExport`** osztályban található a fenti **`order`()** metódus, aminek első paraméterében megadhatjuk egy tömbben a rendelések azonosítóit, második paramétere alapértelmezetten egy üres sztring, ekkor a kimenetre íródik a legenerált XML fájl, amennyiben viszont a 'file' sztringet adjuk meg, a wsesync/exportxml/ mappában készül el a rendeléseket tartalmazó XML fájl. A példában a 86 –os és 87-es azonosítóval rendelkező rendeléseket kérjük le.

A **`WseSyncExport`** osztályban még egy metódust láthatunk, mégpedig a **`maketree`**() statikus metódust. Ennek a függvénynek a segítségével épül fel az adatbázisból lekért rendelési adatok alapján az Orders XML. Lehetőségünk van BASE64 kódolást is beállítani, a példafájlban ez nincs bekapcsolva.

A következő ábrán látható a böngésző kimenetére írt XML, amely a 86-os és a 87-es rendelés adatait tartalmazza. Mivel terjedelmes szöveg íródik a kimenetre a két rendelés közül csak az elsőt jelenítettem meg az ábrán.

5. ábra *Rendelések lekérése XML formátumban*

4. Összegzés

Napjainkban, amikor az internet fénykorát éli, egyre több webáruház jelenik meg a piacon. Az e-kereskedelem területén az utóbbi években radikálisan megnőtt a fogyasztók száma, egyre többen vásárolnak online. A piac növekedése az üzleti alkalmazások fejlődését is elengedhetetlenül magával hozta. Az informatika területén a webes alkalmazások, fejlesztések is középpontba kerültek. Fontos tehát a fejlődéseket követni és ismerni azokat a módszereket, amelyeknek segítségével megkönnyíthetjük nem csak a vásárlók, hanem az eladók és a többi programozó munkáját is.

Szakdolgozatom két nagyobb gondolati egységből áll. Az első részben elméleti oldalról közelítettem meg a szinkronizáció területét. A webáruház és az e-kereskedelem felől kiindulva, a szinkronizálást, mint fogalmat definiáltam. Több féle csoportosítás létezik a szinkronizálásra, ezeket sorba vettem. Irány szerint lehet importálásról és exportálásról beszélni, az idő függvényében létezik szinkron és aszinkron adatátvitel és az automatizáltság foka szerint elkülöníthetünk automatikus, félautomatikus és manuális szinkronizációt.

Azokat a technológiákat, amelyeket felhasználhatunk a szinkronizáció során, vagyis a webszolgáltatásokat részletesebben is megvizsgáltam. A szinkronizálási módszerek felsorolása, és ismertetése mellett, példákat is megemlítettem. Ezeket a módszereket négyféle szempontból összehasonlítottam. Arra a következésre jutottam, hogy nincs legjobb megoldás, kiemelni egyik módszert sem lehet, minden helyzet más-más megoldást kíván. A webáruháztól, de a külső rendszertől is nagy mértékben függ a módszer kiválasztása. Az összehasonlítási szempontokat figyelembe véve (költség, gyorsaság, megbízhatóság, biztonság) kell kiválasztani a megfelelő módszert.

Dolgozatom második nagy fejezetében, egy olyan megoldást mutatok be, amely egy webáruház motorhoz lett készítve és megkönnyíti a külső rendszerekkel történő szinkronizációt. Általános megoldást nyújt egy ügyviteli rendszerrel való automatikus kommunikációhoz az importálás irányában.

Úgy érzem, mindenképp fontos tényezője lehet a jövő webes alkalmazásainak fejlesztése során a szinkronizáció, amely központi témája volt szakdolgozatomnak.

5. Mellékletek

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="WseXml"
  targetNamespace="urn:WseXml"
  xmlns:tns="urn:WseXml"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

    <!-- Típusok definíciója -->
    <types>
        <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:Xml">
            <xsd:element name="getData" type="xsd:string" />
            <xsd:element name="XmlResponse" type="xsd:string" />
        </xsd:schema>
    </types>

    <!-- Üzenetek definíciója -->
    <message name="WseInput">
        <!-- WseInput üzenet -->
        <part name="data" type="tns:getData" />
    </message>
    <message name="WseInputResponse">
        <!-- válaszüzenet a WseInput üzenetre  -->
        <part name="return" type="tns:XmlResponse" />
    </message>

    <!-- Támogatott műveletek -->
    <portType name="XmlPort">
        <!-- WseInput művelet -->
        <operation name="WseInput">
            <input message="tns:WseInput" />
            <output message="tns:WseInputResponse" />
        </operation>
    </portType>

    <!-- Protokoll kötések -->
    <binding name="XmlBinding" type="tns:XmlPort">
        <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
        <operation name="WseInput">
            <soap:operation soapAction="urn:XmlAction" />
            <input>
                <soap:body use="encoded" namespace="urn:Xml"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:Xml"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
    </binding>
```

```
    <!-- Szolgáltatás elérhetősége -->
    <service name="XmlService">
        <port name="XmlPort" binding="tns:XmlBinding">
            <soap:address
location="http://www.sitebuilding.hu/wsesync/wsesync/wsesoapserver0
1.php" />
        </port>
    </service>

</definitions>
```

1. melléklet: wsesoapserver01.wsdl

```
<PRODUCTS>
      <PRODUCT>
         <SKU>sku_0001</SKU>
         <NAME>Teszt termék</NAME>
         <CATEGORY_ID>101,200,201</CATEGORY_ID>
         <SHORT_DESCIRPTION>rövid leírás...</SHORT_DESCIRPTION>
         <LONG_DESCRIPTION>hosszú leírás...</LONG_DESCRIPTION>
         <REGISTRATION_DATE>1269451194</REGISTRATION_DATE>
         <PRICE1>10000</PRICE1>
         <PRICE2>-1</PRICE2>
         <PRICE3>-1</PRICE3>
         <PRICE4>0</PRICE4>
         <PRICE5/>
         <VAT>25</VAT>
         <DISCOUNT>11000</DISCOUNT>
         <UNIT/>
         <STOCK>56</STOCK>
         <IMAGE>teszt01.jpg</IMAGE>
      </PRODUCT>
      <PRODUCT>
         <SKU>sku_0002</SKU>
         <NAME>Egy másik teszt termék</NAME>
         <CATEGORY_ID>101</CATEGORY_ID>
         <SHORT_DESCIRPTION>rövid leírás...</SHORT_DESCIRPTION>
         <LONG_DESCRIPTION>hosszú leírás...</LONG_DESCRIPTION>
         <REGISTRATION_DATE>1269451194</REGISTRATION_DATE>
         <PRICE1>20000</PRICE1>
         <PRICE2>-1</PRICE2>
         <PRICE3>-1</PRICE3>
         <PRICE4>0</PRICE4>
         <PRICE5/>
         <VAT>25</VAT>
         <DISCOUNT/>
         <UNIT/>
         <STOCK>100</STOCK>
         <IMAGE>teszt02.jpg</IMAGE>
         <PARAM1>szín:zöld</PARAM1>
         <PARAM2>méret:xxl</PARAM2>
         <PARAM3>Szállítási idő:1 hét</PARAM3>
      </PRODUCT>
</PRODUCTS>
```

2. melléklet: product.xml

administrator/components/com_wsesync/admin.wsesync.php

administrator/components/com_wsesync/wsesync.config.php

administrator/components/com_wsesync/IO/validation/validation.xml

administrator/components/com_wsesync/IO/validation/product.xml

administrator/components/com_wsesync/IO/validation/category.xml

administrator/components/com_wsesync/IO/validation/stock.xml

administrator/components/com_wsesync/IO/validation/orderstate.xml

administrator/components/com_wsesync/IO/validation/publish.xml

classes/validator/categoryvalidator.class.php

classes/validator/validator.class.php

classes/validator/validatorfactory.class.php

classes/validator/validatorhelper.class.php

classes/wsesync/wsesync.class.php

classes/wsesync/wsesyncexport.class.php

classes/wsesync/wsesynclog.class.php

classes/wsesync/wsesyncplugin.class.php

classes/wsesync/plugin/orderstateplugin.class.php

classes/wsesync/plugin/productplugin.class.php

classes/wsesync/plugin/publishplugin.class.php

classes/wsesync/plugin/stockplugin.class.php

classes/wsesync/plugin/categoryplugin.class.php

classes/xml/xmlobject.class.php

classes/xml/xmlhelper.class.php

classes/xml/xmlhandler.class.php

data/temp/

wsesync/xml_to_obj.php

wsesync/xmlinput.php

wsesync/wsesoapserver01.wsdl

wsesync/wsesoapserver02.wsdl

wsesync/soapcall.php

wsesync/wsesoapserver01.php

wsesync/wsesoapserver02.php

wsesync/wsesync _order_export.php

wsesync/xml/

wsesync/exportxml/

3. melléklet: Azok a fájlok/könyvtárak, amiket a komponens használ.

```
xmlinput.php?type=PRODUCT&xml=<PRODUCTS><PRODUCT><SKU>dXJsUHJvZHVjdDE=</SK
U><NAME>dXJsUHJvZHVjdDE=</NAME><CATEGORY_ID>MQ==</CATEGORY_ID><SHORT_DESCR
IPTION>TG9yZW0gSXBzdW0gaXMgc2ltcGx5IGR1bW15IHRleHQgb2YgdGhlIHByaW50aW5nIGF
uZCB0eXBlc2V0dGluZyBpbmR1c3RyeS4=</SHORT_DESCRIPTION><LONG_DESCRIPTION>TG9
yZW0gSXBzdW0gaXMgc2ltcGx5IGR1bW15IHRleHQgb2YgdGhlIHByaW50aW5nIGFuZCB0eXBlc
2V0dGluZyBpbmR1c3RyeS4=</LONG_DESCRIPTION><REGISTRATION_DATE>MTI2OTQ1MTE5N
A==</REGISTRATION_DATE><PRICE1>OTgwMA==</PRICE1><PRICE2>LTE=</PRICE2><PRIC
E3>LTE=</PRICE3><PRICE4>MA==</PRICE4><PRICE5></PRICE5><VAT>MjU=</VAT><UNIT
>ZGFyYWI=</UNIT><STOCK>MTAw</STOCK></PRODUCT><PRODUCT><SKU>dXJsUHJvZHVjdDI
=</SKU><NAME>dXJsUHJvZHVjdDI=</NAME><CATEGORY_ID>MQ==</CATEGORY_ID><SHORT_
DESCRIPTION>TG9yZW0gSXBzdW0gaXMgc2ltcGx5IGR1bW15IHRleHQgb2YgdGhlIHByaW50aW
5nIGFuZCB0eXBlc2V0dGluZyBpbmR1c3RyeS4=</SHORT_DESCRIPTION><LONG_DESCRIPTIO
N>TG9yZW0gSXBzdW0gaXMgc2ltcGx5IGR1bW15IHRleHQgb2YgdGhlIHByaW50aW5nIGFuZCB0
eXBlc2V0dGluZyBpbmR1c3RyeS4=</LONG_DESCRIPTION><REGISTRATION_DATE>MTI2OTQ1
MTE5NA==</REGISTRATION_DATE><PRICE1>OTgwMA==</PRICE1><PRICE2>LTE=</PRICE2>
<PRICE3>LTE=</PRICE3><PRICE4>MA==</PRICE4><PRICE5></PRICE5><VAT>MjU=</VAT>
<UNIT>ZGFyYWI=</UNIT><STOCK>MTAw</STOCK></PRODUCT></PRODUCTS>
```

4. melléklet: Példa felparaméterezett URL-re.

6. Irodalomjegyzék

[1] Gottdank Tibor: WEBSZOLGÁLTATÁSOK – XML alapú kommunikáció az interneten
Computerbooks, 2005.

[2] Elliotte Rusty Harold: Hatékony XML – 50 bevált módszer XML adatszerkezetek megvalósítására
Kiskapu, 2006.

[3] Talyigás Judit – Mojzes Imre: Az új gazdaság útikönyve: Az elektronikus kereskedelem
Műegyetemi Kiadó, 2004.

[4] Peter Moulding: PHP Fekete könyv
Perfact-Pro Kft., 2002.

[5] Zajdó Csaba: Vevőmágnes Webáruház, 2009.

[6] Sikos László: XHTML – A HTML megújulása XML alapokon
BBS-INFO Kiadó, 2004.

[7] Dr. Adamkó Attila – Webarchitektúrák (Egyetemi jegyzet) 2010.

[8] World Wide Web Consortium (W3C)
http://www.w3.org
2010.04.28.

[9] Miniszterelnöki Hivatal - Elektronikus adatcsere alkalmazása a kormányzatban
http://www.meh.hu/szervezet/hivatalok/ekk/kietb/ajanlasok/adatcsere20040801.html Letölthető anyagok: http://misc.meh.hu/binary/6857_a17.zip 2010.04.28.

[10] ActiveGlobe WebOTX: Business Solution | NEC
http://www.nec.co.jp/middle/WebOTX-e/function/webservice.html
2010.04.28.

[11] Map Middle East: Potential of Using Web Services in Distributed
http://www.gisdevelopment.net/magazine/middleeast/2006/nov-dec/32_2.htm

2010.04.28.

[12] SOAP: Együttműködés távoli szolgáltatásokkal - Weblabor
http://weblabor.hu/cikkek/soap
2010.05.02.

[13] PHP: Hypertext Preprocessor
http://www.php.net
2010.05.02.

Köszönetnyilvánítás

- Köszönetet szeretnék mondani belső témavezetőmnek, dr. Adamkó Attilának, aki elvállalta szakdolgozatom felügyeletét, időt szánt a felmerülő kérdések megvitatására és válaszával segítette munkámat.
- Szeretném megköszönni továbbá Gál Jánosnak, külső témavezetőmnek a segítségét, aki szakmai tudásával, a témában való jártasságával irányt mutatott dolgozatom megírásában.
- Akszenovics Sándor és Kiss Viktor informatikusok támogatása is nagy segítséget jelentett a PHP fejlesztések során felmerült kérdéseimben.
- Köszönet illeti Gottdank Tibort közvetve, hiszen az ő Webszolgáltatások könyve nyújtotta az alapot dolgozatomhoz. A könyv annak ellenére, hogy 2003-ban jelent meg először, még mindig nagyon sok hasznos információt hordoz a webszolgáltatások témakörében.
- Szeretném szüleimnek megköszönni a támogatást, amit tanulmányaim során kaptam tőlük.
- Köszönöm barátnőmnek, Pallagi Nikolettának, hogy a szakdolgozat írása alatt támogatott és legfőbb kritikusom volt a megfogalmazások terén.
- Továbbá köszönet illeti barátaimat, csoporttársaimat, akik az egyetemi évek alatt és szakdolgozatom írása során is támaszt nyújtottak.
- Végül a tanulmányaim során azoknak a tanároknak szeretném megköszönni a oktatást, akik hozzájárultak ahhoz, hogy ma egy ilyen szakmai munkát adjak ki a kezemből.