NDA tulajdonnévtér
Fejlesztési dokumentáció

Megbízó: Neumann János Digitális Könyvtár és Multimédia Központ Kht.
Nemzeti Digitális Adattár Szakmai Igazgatóság

Tartalom

1. Forrásfájlok

1.1. addevent.php

Ez a fájl tartalmazza a névadatok szerkesztésénél új esemény felvitelére szolgáló űrlapot megjelenítő kódot. Ezt a fájlt az eventhandling2.js fájlban definiált addEvent(ndanameid,ndaobjectid) JavaScript függvény hívja meg új ablakban. Az ndanameid és ndaobjectid függvényparaméterek határozzák meg, hogy melyik névhez illetve névhordozóhoz vesszük fel az új eseményt. Mentésnél az űrlapon megadott adatok POST metódossal adódnak át az addevent2.php fájlnak.

1.2. addevent2.php

Ez a fájl tartalmazza a névadatok szerkesztésénél új esemény felvitelét végző kódot. Ezt a fájlt az addevent.php fájl hívja meg, $_POST változóban kapja meg az új esemény adatait, ami alapján összeállítja és meghívja az új esemény felvitelét végző XML formátumú NDA-protokoll kérést. A kérésre érkező válasz XML-t feldolgozza. Hiba esetén kiírja a hibaüzenetet, egyébként bezárja az ablakot és frissíti a szülő ablakban az eseményadatokat megjelenítő listát.

1.3. addnewname1.php

Az addnewname1.php tartalmazza az új név felvitel első lépésének forráskódját. A program a „logedin” munkament változó értékének kiolvasásával leellenőrzi, hogy bejelentkezett felhasználó hívta-e meg ezt a funkciót. Ha nem, akkor a bejelentkező oldal jelenik meg.
A chooseName() javascript függvény nyitja meg új ablakban a névkiválasztó keresőt névváltozat felviteléhez. A kiválasztott név és névhordozójának azonosítói rejtett űrlapmezőkbe kerülnek és a névtértípus értékével együtt POST metódussal adódnak át a következő (addnewname2.php) oldalra.

1.4. addnewname2.js

Ez a fájl tartalmazza azoknak a JavaScript függvényeknek a forráskódját, amelyek az új név felvitelnél és szerkesztésnél az adatbeviteli űrlap különleges funkcióit irányítják.
A fontosabb függvények rövid leírása:
makeRequest(url,div): Ez a függvény indítja az aszinkron http kéréseket. A függvény az url paraméterben adott URL-t hívja meg. A letöltött adatok megjelenítését a displayContents függvény végzi. A div paraméter értékét ennek a függvénynek adjuk át.
displayContents(http_request,div): Ez a függvény végzi a makeRequest függvény által indított kérés válaszaként letöltött adatok megjelenítését, amit a div paraméterben adott <DIV> HTML elembe ír.
addContext(): Ez a függvény végzi az új kontextus felvételét.
remContext(index): Ez a függvény végzi az index paraméter által meghatározott kontextus törlését a kontextus listából.
writeContexts(): Ez a függvény jeleníti meg a kiválasztott kontextusokat, és hozza létre a kontextusok adatainak továbbítására szolgáló rejtett űrlapmezőket.
addProf(): Ez a függvény végzi az új foglalkozás felvételét.
function remProf(index) : Ez a függvény végzi az index paraméter által meghatározott foglalkozás törlését a foglalkozás listából.
writeProfs(): Ez a függvény jeleníti meg a kiválasztott foglalkozásokat, és hozza létre a foglalkozások adatainak továbbítására szolgáló rejtett űrlapmezőket.
addFootprint(): Ez a függvény végzi a geometriai adatoknál egy új pont felvételét.

remFootprint(index) : Ez a függvény végzi geometriai adatoknál az index paraméter által meghatározott pont törlését a listából.
writeFootprint(): Ez a függvény jeleníti meg a geometriai adatoknál a pontok földrajzi koordinátáit, és hozza létre a pontok adatainak továbbítására szolgáló rejtett űrlapmezőket.

1.5. addnewname2.php

Az addnewname2.php tartalmazza az új név felvitel második lépésének forráskódját. A program a „logedin" munkament változó értékének kiolvasásával leellenőrzi, hogy bejelentkezett felhasználó hívta-e meg ezt a funkciót. Ha nem, akkor a bejelentkező oldal jelenik meg. A program attól függően jeleníti meg a névfelviteli űrlapot, hogy az addnewname1.php oldalon milyen névtértípust választottunk. Ha érkezik ndanameid és ndaobjectid, akkor névváltozatként vesszük fel az új nevet. Ekkor lekérdezzük, és az űrlapon megjelenítjük a névhordozó összes adatát: személyeknél a nemet, földrajzi objektumoknál a geometriai adatokat. A névhordozó adatait az NDA-protokoll szerint XML szerkezetű kéréssel kérdezzük le, és a válasz is XML-ben érkezik. A válasz XML feldolgozása a PHP DOM csomagjának függvényei segítségével történik. A név adatainak megadására szolgáló űrlap forráskódja azonos a felvitelnél és a szerkesztésnél, ezért azt a nameform.php tartalmazza és az include PHP függvénnyel illesztjük be.
Az űrlapon lévő eseménytípus és kapcsolattípus-választó lenyíló listák tartalma az adatbázisban tárolt típusértékeket tartalmazza, melyeket az NDA-protokoll segítségével kérdezzük le.
Mentéskor az addnewname3.php fájlt hívjuk meg és a név felvételekor megadott adatokat POST metódussal küldi el a program.

1.6. addnewname3.php

Ez a fájl tartalmazza az új név felvitel harmadik lépésének forráskódját. Ebben a lépésben történik az addnewname2.php oldalon megadott adatok mentése. Az elmentendő adatok a $_POST PHP változóban érkeznek. A harmadik lépésben, ha nem névváltozatot viszünk fel, akkor a program meghívja az új névhordozó felvitelét végző NDA-protokoll kérést. Ha ez sikerült vagy meglévő névhordozóhoz veszünk fel új névváltozatot, akkor a program összeállítja és meghívja az új név felvitelét végző NDA-protokoll kérést. Ha olyan nevet viszünk fel, amely hasonlít egy már meglévőhöz, akkor a program figyelmeztet erre, kilistázza a hasonló neveket, és megerősítést kér a név felvételére vonatkozóan. Ha a felhasználó megerősíti, hogy valóban fel akarja vinni a nevet, akkor meghívódik az addnewname4.php. Ekkor a $_POST változó értékét egy munkamenet változóba töltjük.

1.7. addnewname4.php

Ez a fájl tartalmazza az új név felvitel negyedik lépésének forráskódját. A névfelvitel negyedik lépésére, akkor van szükség, ha a felvinni kívánt névhez volt hasonló a rendszerben. A program összeállítja és meghívja a név felvitelének megerősítését végző NDA-protokoll kérést.

1.8. addnewnameplus.php

Ez a fájl tartalmazza annak a programnak a forráskódját, amely az új név felvitelénél az események, az eseményekhez kapcsolódó nevek, a kapcsolódó objektumok és a megjegyzések felvitelét végző NDA-protokoll kéréseket összeállítja és meghívja. Mivel ezt a megerősítéses és az anélküli esetben is használjuk ezért az addnewnameplus.php fájlt az addnewname3.php és az addnewname4.php fájlba is beillesztjük a PHP include függvényével.

1.9. addnewname_nameparts.php

Ez a fájl az új személynév felvitelekor történő névelemekre bontás kódját tartalmazza. A megadott személynév alapján elvégzi annak felbontását és visszaírja a felbontott nevet az űrlapba.

1.10. addrelobj1.php

Ez a fájl tartalmazza a névadatok szerkesztésénél, új kapcsolódó objektum felvitel első lépésének forráskódját, ahol a kapcsolattípust lehet kiválasztani. Ezt a fájlt az eventhandling2.js fájlban definiált addRelatedObject (ndaobjectid) JavaScript függvény hívja meg új ablakban. Az ndaobjectid függvényparaméter határozza meg, hogy melyik névhordozóhoz vesszük fel az új kapcsolódó objektumot. A kiválasztott kapcsolattípus POST metódussal adódik át a search2choose1.php fájlnak.

1.11. admincomments.php

Ennek a fájlnak a segítségével lehet megtekinteni az adatgazdák által küldött észrevételeket.

1.12. adminDuplums.php

Ennek a féjlnak a segítségével lehet megtekinteni a rendszer által duplumgyanúsnak vélt neveket. A kilistázott nevek egyikére kattintva a program megkeresi a névalak, születési hely és idő alapján a rendszerben található hasonló neveket. Amennyiben az adminisztrátor úgy véli, hogy a kiváalsztott név valamelyik, a találati listában szereplő névvel azonos névhordozóhoz tartozik, akkor lehetősége van összevonni a neveket.

1.13. adminFunctions.php

Ez a fájl tartalmazza a felhasználókezeléssel kapcsolatos összes adminfunkciót. Ezzel lehet megtekinteni az adminisztrátorok és az adatgazdák listáját, fel lehet venni új adminisztrátort, vagy adatgazdát, illetve kezelni lehet az adatgazdák regisztrációit.

1.14. adminlogin.php

Ez a fájl tartalmazza az adminisztrátorok beléptetését végző program forráskódját. A belépő űrlapon megadott felhasználónév és jelszó alapján a program előállítja és meghívja a belépést végző NDA-protokoll kérést. Sikeres bejelentkezés esetén PHP munkamenet változóban tároljuk a belépés tényét, az NDA-protokoll által generált virtuális munkamenet azonosítót, a felhasználó azonosítót, és azt, hogy adminisztrátorként léptünk be.

1.15. csvUpload.php

Ennek a fájlnak a segítségével tölthet be az adatgazda az adtbázisba csv kiterjesztésű fájlban tárolt neveket. A névtértípus kiválasztása után lehetőség van a megfelelő formátumú csv fájl feltöltésére. A feltöltés után a program soronként viszi fel az adatokat az adatbázisba, majd egy fájlba írja vissza a felvitt adatokat, illetve az esetlegesen előforduló hibákat.

1.16. dataOwner.blank

Az adatgazdák adatainak megjelenítéséhez szükséges HTML sémát tartalmazza.

1.17. delevent.php

Ez a fájl tartalmazza a névadatok szerkesztésénél az eseménytörlés első lépését végző program forráskódját. Ezt a fájlt az eventhandling2.js fájlban definiált remEvent(eventid,

ndaid) JavaScript függvény hívja meg új ablakban. Az eventid és ndaid függvényparaméterek határozzák meg, hogy melyik eseményt kell törölni. Megerősítésnél a törlendő esemény adatai GET változóban adódnak át a delevent2.php fájlnak.

1.18. delevent2.php

Ez a fájl tartalmazza a névadatok szerkesztésénél az eseménytörlés második lépését végző program forráskódját. Ezt a fájlt a delevent.php fájl hívja meg, és $_GET változóban kapja meg a törlendő esemény adatait, ami alapján összeállítja és meghívja az esemény törlését végző XML formátumú NDA-protokoll kérést. A kérésre érkező válasz XML-t feldolgozza. Hiba esetén kiírja a hibaüzenetet, egyébként bezárja az ablakot és frissíti a szülő ablakban az eseményadatokat megjelenítő listát.

1.19. deleventrelobj.php

Ez a fájl tartalmazza a névadatok szerkesztésénél az eseményhez kapcsolódó név törlésének első lépését végző program forráskódját. Ezt a fájlt az eventhandling2.js fájlban definiált remEventRelatedObject(eventid, ndanameid) JavaScript függvény hívja meg új ablakban. Az eventid és ndanameid függvényparaméterek határozzák meg, hogy melyik nevet, melyik eseménytől kell törölni. Megerősítésnél a törlendő név adatai GET változóban adódnak át a deleventrelobj 2.php fájlnak.

1.20. deleventrelobj2.php

Ez a fájl tartalmazza a névadatok szerkesztésénél az eseményhez kapcsolódó név törlésének második lépését végző program forráskódját. Ezt a fájlt a deleventrelobj.php fájl hívja meg, és $_GET változóban kapja meg a törlendő név adatait, ami alapján összeállítja és meghívja az név törlését végző XML formátumú NDA-protokoll kérést. A kérésre érkező válasz XML-t feldolgozza. Hiba esetén kiírja a hibaüzenetet, egyébként bezárja az ablakot és frissíti a szülő ablakban az eseményadatokat megjelenítő listát.

1.21. delnote.php

Ez a fájl tartalmazza a névadatok szerkesztésénél a megjegyzések törlésének első lépését végző program forráskódját. Ezt a fájlt az eventhandling2.js fájlban definiált remNote(ndaid, noteid) JavaScript függvény hívja meg új ablakban. Az ndaid és noteid függvényparaméterek határozzák meg, hogy melyik megjegyzést kell törölni. Megerősítésnél a törlendő megjegyzés adatai GET változóban adódnak át a delnote2.php fájlnak.

1.22. delnote2.php

Ez a fájl tartalmazza a névadatok szerkesztésénél a megjegyzés törlésének második lépését végző program forráskódját. Ezt a fájlt a delnote.php fájl hívja meg, és $_GET változóban kapja meg a törlendő megjegyzés adatait, ami alapján összeállítja és meghívja az megjegyzés törlését végző XML formátumú NDA-protokoll kérést. A kérésre érkező válasz XML-t feldolgozza. Hiba esetén kiírja a hibaüzenetet, egyébként bezárja az ablakot és frissíti a szülő ablakban a megjegyzésadatokat megjelenítő listát.

1.23. delrelobj.php

Ez a fájl tartalmazza a névadatok szerkesztésénél a kapcsolódó objektum törlésének első lépését végző program forráskódját. Ezt a fájlt az eventhandling2.js fájlban definiált remRelatedObject(leftndaobjectid, relationtypeid, rightndaobjectid) JavaScript függvény hívja meg új ablakban. Az leftndaobjectid, relationtypeid és rightndaobjectid függvényparaméterek

határozzák meg, hogy melyik kapcsolódó objektumot kell törölni. Megerősítésnél a törlendő kapcsolódó objektum adatai GET változóban adódnak át a delrelobj2.php fájlnak.

1.24. delrelobj2.php

Ez a fájl tartalmazza a névadatok szerkesztésénél a kapcsolódó objektum törlésének második lépését végző program forráskódját. Ezt a fájlt a delrelobj.php fájl hívja meg, és $_GET változóban kapja meg a törlendő kapcsolódó objektum adatait, ami alapján összeállítja és meghívja a kapcsolódó objektum törlését végző XML formátumú NDA-protokoll kérést. A kérésre érkező válasz XML-t feldolgozza. Hiba esetén kiírja a hibaüzenetet, egyébként bezárja az ablakot és frissíti a szülő ablakban a kapcsolódó objektumok adatait megjelenítő listát.

1.25. editevent.php

Ez a fájl tartalmazza a névadatok szerkesztésénél, meglévő esemény szerkesztésének első lépését végző program forráskódját. Ezt a fájlt az eventhandling2.js fájlban definiált editEvent(eventid) JavaScript függvény hívja meg új ablakban. Az eventid függvényparaméter határozza meg, hogy melyik esemény adatait kell szerkeszteni. A szerkesztendő esemény adatait egy XML formátumú NDA-protokoll kéréssel kérdezzük le. A válasz szintén XML formátumban érkezik, amit a PHP DOM függvényeivel dolgozunk fel. A szerkesztett esemény adatai POST metódussal adódnak át az editevent2.php fájlnak.

1.26. editevent2.php

Ez a fájl tartalmazza a névadatok szerkesztésénél, meglévő esemény szerkesztésének második lépését végző program forráskódját. Ezt a fájlt az editevent.php fájl hívja meg, és $_POST változóban kapja meg a szerkesztett esemény adatait, ami alapján összeállítja és meghívja az eseményadatok módosítását végző XML formátumú NDA-protokoll kérést. A kérésre érkező válasz XML-t feldolgozza. Hiba esetén kiírja a hibaüzenetet, egyébként bezárja az ablakot és frissíti a szülő ablakban az események adatait megjelenítő listát.

1.27. editname2.php

Ez a fájl tartalmazza a névadatok szerkesztésének, első lépését végző program forráskódját. A szerkesztendő névhez tartozó ndanameid és ndaobjectid GET paraméterben érkezik. A program összeállítja és meghívja a név és a névhordozó adatait lekérő XML formátumú NDA-protokoll kéréseket. A válaszok szintén XML formátumban érkeznek, amit a PHP DOM függvényeivel dolgozunk fel, abból a célból, hogy az aktuális adatok megjelenjenek a szerkesztő űrlapon. A módosított adatok POST metódussal adódnak át az editname3.php fájlnak.

1.28. editname3.php

Ez a fájl tartalmazza a névadatok szerkesztésének, második lépését végző program forráskódját. Ezt a fájlt az editname2.php fájl hívja meg, és $_POST változóban kapja meg a szerkesztett név adatait, ami alapján összeállítja és meghívja a módosítást végző XML formátumú NDA-protokoll kéréseket. A kérésekre érkező válasz XML-eket feldolgozza és hiba esetén kiírja a hibaüzenetet.

1.29. editname_nameparts.php

Ez a fájl a személynév szerkesztésekor történő névelemekre bontás kódját tartalmazza. A megadott személynév alapján elvégzi annak felbontását, elmenti az új felbontást és visszaírja a felbontott nevet az űrlapba.

1.30. editnote.php

Ez a fájl tartalmazza a névadatok szerkesztésénél, meglévő megjegyzés szerkesztésének első lépését végző program forráskódját. Ezt a fájlt az eventhandling2.js fájlban definiált editNote(ndaid, noteid) JavaScript függvény hívja meg új ablakban. Az ndaid és az noteid függvényparaméter határozza meg, hogy melyik megjegyzés adatait kell szerkeszteni. A szerkesztendő megjegyzés adatait egy XML formátumú NDA-protokoll kéréssel kérdezzük le. A válasz szintén XML formátumban érkezik, amit a PHP DOM függvényeivel dolgozunk fel. A szerkesztett megjegyzés adatai POST metódussal adódnak át az editnote2.php fájlnak.

1.31. editnote2.php

Ez a fájl tartalmazza a névadatok szerkesztésénél, meglévő megjegyzés szerkesztésének második lépését végző program forráskódját. Ezt a fájlt az editnote.php fájl hívja meg, és $_POST változóban kapja meg a szerkesztett megjegyzés adatait, ami alapján összeállítja és meghívja a megjegyzésadatok módosítását végző XML formátumú NDA-protokoll kérést. A kérésre érkező válasz XML-t feldolgozza. Hiba esetén kiírja a hibaüzenetet, egyébként bezárja az ablakot és frissíti a szülő ablakban az megjegyzések adatait megjelenítő listát.

1.32. editrule.php

Ennek a fájnak a segítségével szerkeszthetjük a névelemrendezési szabályokat. A szerkeszteni kívánt szabály kiválasztása után lehetőség van a szabály szerinti névelem típusok sorrendjének megváltoztatására. A program a típuslistaszerkesztőből hívható meg.

1.33. eventhandling.js

Ez a fájl tartalmazza azoknak a JavaScript függvényeknek a forráskódját, amelyek az új név felvitelnél az adatbeviteli űrlap különleges funkcióit irányítják.
A fontosabb függvények rövid leírása:
addEvent(): Ez a függvény végzi az új esemény felvételét.
remEvent(index): Ez a függvény végzi az index paraméter által meghatározott esemény törlését az esemény listából.
addEventRelatedObject(namestr, ndanameid): Ez a függvény végzi új kapcsolódó név felvételét eseményhez. A namestr paraméterben a kapcsolódó nevet, az ndanameid paraméterben pedig annak azonosítóját adjuk át.
remEventRelatedObject(index): Ez a függvény végzi az index paraméter által meghatározott eseményhez kapcsolódó név és az esemény közti kapcsolat törlését.
writeEventRelatedObjects(): Ez a függvény jeleníti meg az egyes eseményekhez kapcsolódó neveket, és hozza létre azok adatainak továbbítására szolgáló rejtett űrlapmezőket.
writeEvents(): Ez a függvény jeleníti meg az eseményeket, és hozza létre az események adatainak továbbítására szolgáló rejtett űrlapmezőket.
addRelatedObject(namestr, ndaobjectid): Ez a függvény végzi új kapcsolódó objektum felvételét. A namestr paraméterben a kapcsolódó objektum nevét, az ndaobjectid paraméterben pedig a névhordozó azonosítóját adjuk át.
remRelatedObject(index) : Ez a függvény végzi az index paraméter által meghatározott kapcsolódó objektum törlését a kapcsolódó objektumok listájából.
writeRelatedObjects(): Ez a függvény jeleníti meg a kapcsolódó objektumokat, és hozza létre azok adatainak továbbítására szolgáló rejtett űrlapmezőket.
addNote(): Ez a függvény végzi az új megjegyzés felvételét.
remNote(index) : Ez a függvény végzi az index paraméter által meghatározott megjegyzés törlését az esemény listából.

writeNotes(): Ez a függvény jeleníti meg a megjegyzéseket, és hozza létre a megjegyzések adatainak továbbítására szolgáló rejtett űrlapmezőket.

1.34. eventhandling2.js

Ez a fájl tartalmazza azoknak a JavaScript függvényeknek a forráskódját, amelyek a névszerkesztésnél az adatbeviteli űrlap különleges funkcióit irányítják.
A fájlban definiált függvények és paramétereik:
addEvent(ndanameid,ndaobjectid)
editEvent(eventid)
remEvent(eventid, ndaid)
addEventRelatedObject(eventid)
remEventRelatedObject(eventid, ndanameid)
addRelatedObject(ndaobjectid)
remRelatedObject(leftndaobjectid, relationtypeid, rightndaobjectid)
addNote(ndanameid, ndaobjectid)
editNote(ndaid, noteid)
remNote(ndaid, noteid)
A függvények közös jellemzője, hogy felugró ablakban meghívnak egy PHP fájlt, amelynek lekérdező karakterláncban (GET paraméterben) átadják az argumentumukban lévő paramétereket. A meghívott fájlok leírásában szerepel, hogy melyik függvény hívja meg.

1.35. extendedSearch.php

Az extendedSearch.php tartalmazza a részletes kereső forráskódját. A kiválasztott névtértípusnak megfelelő tulajdonságok alapján a program összeállítja a keresési űrapot. Az űrlapon kitöltött értékek alapján a program elküldi a megfelelő XML kérést a kérésértelmezőnek, majd a keresés eredménye a megfelelő lapozási információkkal együtt megjelenik az ürlap után.

1.36. feedback.php

Ennek a fájlnak a segítségével tudja az adatgazda a webes felület használata közben felmerült észrevételeit, ötleteit eljuttatni az adminisztrátornak. A fájl által megjelenített űrlapon megadott észrevétel, mentés után bekerül az adatbázisba, ahonnan később névtér-adminisztrátor kilistázhatja.

1.37. functions.inc

Ez a fájl tartalmazza a rendszer használatához szükséges függvényeket.
A függvények neve, paraméterei és működése:
db_query($query): A függvény végrehajtja a $query paraméterben kapott SQL lekérdezést. Ezt több lépésben teszi. Először létrehozza a kapcsolatot az adatbázis-kezelővel, végrehajtja a lekérdezést, véglegesíti a lekérdezés eredményét majd bontja a kapcsolatot. Mivel Oracle adatbázis-kezelővel dolgozunk, ezért a PHP beépített (oci) függvényeit használjuk. Sikeres lekérdezés esetén az adatok elérésére alkalmas $result változóval, ellenkező esetben hamis értékkel tér vissza a függvény.
connect_to_db(): A függvény csatlakozik az Oracle adatbázis-kezelőhöz és visszatér a csatlakozás eredményével.
nc_query($connection, $queryString): A függvény végrehajtja a $queryString paraméterben kapott adatbázis lekérdezést a $connection kapcsolódással.

end_trans($connection,$needCommit,$queryArr=array()): A függvény befejezi az adatbázis tranzakciót. Ha a $needCommit paraméter értéke hamis, akkor rollback műveletet hajt végre, vagyis visszavonja a függő lekérdezéseket.
nc_delete($connection,$prc): A függvény végrehajt egy rekord törlést és visszatér a törölt rekordban lévő azonosítóval, ha sikerült. Ellenkező esetben hamissal. A törlés nem véglegesítődik, visszavonásra kerül.
db_delete($prc): A függvény végrehajt egy rekord törlést és visszatér a törölt rekordban lévő azonosítóval, ha sikerült. Ellenkező esetben hamissal. A törlés véglegesítődik.
nc_insert($query): A függvény végrehajt egy rekord beszúrást és visszatér a beszúrt rekordban lévő azonosítóval, ha sikerült. Ellenkező esetben hamissal. A beszúrás nem véglegesítődik, visszavonásra kerül.
db_insert($query): A függvény végrehajt egy rekord beszúrást és visszatér a beszúrt rekordban lévő azonosítóval, ha sikerült. Ellenkező esetben hamissal. A beszúrás véglegesítődik.
generate_password($length): A függvény generál egy $length hosszú karakterláncot, amelyben véletlenszerűen csak az angol ábécé kis és nagybetűi, illetve számok szerepelnek.
send_allowmail($username,$name,$email,$password): Ez a függvény küldi el az adatgazda engedélyezéséről szóló értesítő levelet. Az üzenet tartalmazza a bejelentkezéshez szükséges jelszót, amit a függvény a $password paraméterben kap.
send_denymail($username,$name,$email): Ez a függvény küldi el az adatgazda regisztrációjának eluasításáról szóló értesítő levelet.
send_reallowmail($username,$email): Ez a függvény küldi el a letiltott adatgazda újraengedélyezéséről szóló értesítő levelet.
send_blockmail($username,$email): Ez a függvény küldi el az adatgazda letiltásáról szóló értesítő levelet.
count_seconds_dayprecision($pre,$year,$month,$day): Ez a függvény számolja ki, hogy a paraméterek által meghatározott nap első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_monthprecision($pre,$year,$month): Ez a függvény számolja ki, hogy a paraméterek által meghatározott hónap első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_yearprecision($pre,$year): Ez a függvény számolja ki, hogy a paraméterek által meghatározott év első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_decadeprecision($pre,$decade): Ez a függvény számolja ki, hogy a paraméterek által meghatározott évtized első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_centuryprecision($pre,$century): Ez a függvény számolja ki, hogy a paraméterek által meghatározott évszázad első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_millenaryprecision($pre,$millenary): Ez a függvény számolja ki, hogy a paraméterek által meghatározott évezred első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_betweenyears($pre,$startyear,$stopyear): Ez a függvény számolja ki, hogy a paraméterek által meghatározott időintervallum kezdőévének első és utolsó évének utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_betweencenturies($pre,$startcentury,$stopcentury): Ez a függvény számolja ki, hogy a paraméterek által meghatározott időintervallum kezdőévszázadának első és utolsó évszázadának utolsó másodperce hányadik időszámításunk kezdetéhez képest.

count_seconds_halfofcentury($pre,$century,$half): Ez a függvény számolja ki, hogy a paraméterek által meghatározott fél évszázad első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_aroundyear($pre,$year): Ez a függvény számolja ki, hogy a paraméterek által meghatározott év körüli időszak első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_beforeafteryear($pre,$year,$beforeafter): : Ez a függvény számolja ki, hogy a paraméterek által meghatározott év előtti vagy utáni időszak első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_pieceofdecade($pre,$decade,$piece): Ez a függvény számolja ki, hogy a paraméterek által meghatározott évtized egy szakaszának első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_pieceofcentury($pre,$century,$piece): Ez a függvény számolja ki, hogy a paraméterek által meghatározott évszázad egy szakaszának első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
count_seconds_pieceofmillenary($pre,$millenary,$piece): Ez a függvény számolja ki, hogy a paraméterek által meghatározott évezred egy szakaszának első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
roman2arabic($roman): A függvény római számot arab számmá alakít.
time_handling($datum): A függvény egy szöveges formában adott időadatra kiszámolja, hogy első és utolsó másodperce hányadik időszámításunk kezdetéhez képest.
GetChildren($vals, &$i): A GetXMLTree segédfüggvénye, amely az XML fa szerkezetű reprezentációjában egy csomópont alatti részt visszaadja.
GetXMLTree($xmlstr,$cap=1): Ez a függvény egy XML formátumú szöveget olyan asszociatív tömbbé alakít, amely az XML fa szerkezetű ábrázolása.

1.38. getevents.php

Ez a fájl tartalmazza a névadatok szerkesztésénél, az események adatait kilistázó program forráskódját. A program a $_GET változóban kapott ndaobjectid és ndanameid alapján előállítja és meghívja a névhez és a névhordozóhoz tartozó események adatait lekérdező XML formátumú NDA-protokoll kéréseket. Az XML formátumú válaszok alapján pedig előállítja az események adatait megjelenítő HTML kódot.

1.39. getnotes.php

Ez a fájl tartalmazza a névadatok szerkesztésénél, a megjegyzések adatait kilistázó program forráskódját. A program a $_GET változóban kapott ndaobjectid és ndanameid alapján előállítja és meghívja a névhez és a névhordozóhoz tartozó megjegyzések adatait lekérdező XML formátumú NDA-protokoll kéréseket. Az XML formátumú válaszok alapján pedig előállítja az megjegyzések adatait megjelenítő HTML kódot.

1.40. getrelobjects.php

Ez a fájl tartalmazza a névadatok szerkesztésénél, a kapcsolódó objektumok adatait kilistázó program forráskódját. A program a $_GET változóban kapott ndaobjectid alapján előállítja és meghívja a névhordozóhoz tartozó kapcsolódó objektumok adatait lekérdező XML formátumú NDA-protokoll kéréseket. Az XML formátumú válaszok alapján pedig előállítja a kapcsolódó objektumok adatait megjelenítő HTML kódot.

1.41. getRequests.php

Ez a fájl tartalmazza a módosítási kérések adatait kilistázó program forráskódját. A program a $_GET változóban kapott modListNameID és modListObjectID alapján előállítja és meghívja a névhez és a névhordozóhoz tartozó módosítási kérések adatait lekérdező XML formátumú NDA-protokoll kéréseket. Az XML formátumú válaszok alapján pedig előállítja a módosítási kérések adatait megjelenítő HTML kódot. Szintén ez a program végzi a módosítási kérések elutasítását, illetve elfogadását, mely szintén a megfelelő NDA-protokoll kérések előállításával és meghívásával történik.

1.42. getState.php

Ez a fájl tartalmazza az adatlap korábbi állapotait megjelenítő program forráskódját. A program a $_GET változóban kapott stateListNameID és stateListObjectID alapján előállítja és meghívja az adatlap korábbi állapotait lekérdező XML formátumú NDA-protokoll kéréseket. Az XML formátumú válaszok alapján pedig előállítja az adatlap korábbi állapotait megjelenítő HTML kódot.

1.43. help.php

Ez a fájl jeleníti meg az oldal használatához segítséget nyújtó leírásokat a hivatkozott oldaltól függően.

1.44. history.htm

A névadatok történetének megjelenítéséhez szükséges HTML sémát tartalmazza.

1.45. historyState.htm

A névadatok egy korábbi állapotának megjelenítéséhez szükséges HTML sémát tartalmazza.

1.46. impresszum.php

Ebben a fájlban található az Impresszum menüpont tartalmi része.

1.47. index.php

Az index.php tartalmazza a nyitóoldal forráskódját, ami a fejléc, a menüsor, a keresődobozok és a lábléc megjelenítését végzi. A fejléc, a menüsor, és a lábléc HTML forrásai a include/webinterface.ini fájlban találhatók. A menüsor tartalma attól függő, hogy be van-e jelentkezve a felhasználó, és ha igen, akkor adminisztrátorként, vagy adatgazdaként. Ennek a vizsgálatát az index.php végzi. Itt történik annak a vizsgálata is, hogy a keresőoldalt a közadat keresőből hívták-e meg. Ha igen, akkor egy munkamenet változó beállításával jelezzük, hogy a keresési találati listában az egyes elemeket ki lehet választani úgy, hogy az adatai átadódnak a közadat keresőnek.

1.48. jogi_nyilatkozat.php

Ebben a fájlban található a Jogi nyilatkozat menüpont tartalmi része.

1.49. kapcsolat.php

Ebben a fájlban találhat a Kapcsolat menüpont tartalmi része.

1.50. keresertelmezo3k2.php

Ez a fájl tartalmazza a processRequest($requestxml,$toweb=0) függvény forráskódját, amely az XML formátumú NDA-protokoll kérések feldolgozását végzi. Ha a $toweb paraméter

értéke 0, akkor a válasz XML-t a kimenetre küldi a program, ha ez az érték 1, akkor visszatér a válasz XML szövegével a függvény. A függvény a kérésben érkező adatokból összeállít egy paramétertömböt, amely átadódik a különböző kéréseket végrehajtó segédfájloknak. Ezek a segédfájlok a „keresertelmezo” könyvtárban találhatók. A segédfájlokban található kódok végzik a szükséges adatbázisműveleteket és állítják elő az XML formátumú választ.

1.51. login.php

Ez a fájl végzi az adatgadák bejelentkezését. Sikeres belépés esetén a megfelelő SESSION változók beállításra kerülnek.

1.52. logout.php

Ez a fájl végzi az adatgadák, illetve az adminisztrátorok kijelentkezését. A program törli az összes SESSION változót.

1.53. modRequest.htm

A névadatok módosítási listájának megjelenítéséhez szükséges HTML sémát tartalmazza.

1.54. nameform.php

Ez a fájl tartalmazza a név adatainak megadására szolgáló űrlap forráskódját. Az űrlapon lévő névtípus és kontextusválasztó lenyíló listák tartalma az adatbázisban tárolt típusértékeket tartalmazza, melyeket az NDA-protokoll segítségével kérdezzük le.

1.55. nda.css

Ez a fájl tartalmazza a webes felület stílusbeállításait.

1.56. ownerData.php

Ez a fájl jeleníti meg az egyes rekordok tulajdonosának az adatlapját.

1.57. redirect.php

Ez a fájl naplózza azt az esetet, amikor egy adatgazda adatapján a hozzá tartozó weboldalra kattintanak.

1.58. refreshcontent2.js

Ez a fájl tartalmazza azoknak a JavaScript függvényeknek a forráskódját, amelyek a névszerkesztésnél az adatbeviteli űrlap különleges funkcióit irányítják.
A fájlban definiált függvények, paramétereik és rövid leírásuk:
searchNames(searchtext, searchtype, objecttypeid, pagenumber, itemperpage, divid): Ez a függvény végzi a keresési lista egy lapjának megjelenítését. A függvény aszinkron http kéréssel meghívja a searchresult.php fájlt melynek POST metódussal átadja az argumentumban érkező keresési paramétereket.
showDatasheet(ndaobjectid,ndanameid,divid): Ez a függvény végzi egy névtérelem adatlapjának megjelenítését. A függvény aszinkron http kéréssel meghívja a webinterface5.php fájlt melynek POST metódussal átadja az argumentumban érkező paramétereket.
showResult(http_request,divid), displayContents(http_request,div): A függvény az aktuális http kérésre válaszként érkező tartalmat a divid paraméter által meghatározott div elembe írja.

makeRequest(url,div): A függvény aszinkron http kéréssel meghívja az url paraméterben adott fájlt és a showResult függvény segítségével a válaszként érkező tartalmat a divid paraméter által meghatározott div elembe írja.
changeVisibility(div): Ez a függvény a div paraméterben adott div elem láthatóságát az ellenkezőjére állítja.

1.59. registration.php

Ezen a fájlon keresztül tud egy adatgazda regisztrálni a rendszerbe. Ha az adatgazda bejelentkezve hívja meg ezt az oldalt, akkor megváltoztathatja a regisztrációkor adott adatait.

1.60. request.htm

A névadatok módosítási javaslatának megjelenítéséhez szükséges HTML sémát tartalmazza.

1.61. search2choose1.php

Ez a fájl tartalmazza a névadatok felvitelénél, illetve szerkesztésénél a névkiválasztás első lépését végző program forráskódját. Ebben a lépésben lehet kiválasztani, hogy melyik névtérből akarunk nevet választani. Ha kapcsolódó objektumot akarunk választani, akkor a kapcsolattípusnak megfelelően automatikusan kiválasztódik a névtértípus. A keresőmezőbe írt kifejezésnek megfelelő találati listát a searchNames2Choose JavaScript függvény jeleníti meg.

1.62. search2choose2.php

Ez a fájl tartalmazza a névadatok felvitelénél, illetve szerkesztésénél a névkiválasztás második lépését végző program forráskódját. Attól függően, hogy kapcsolódó objektum, vagy eseményhez kapcsolódó név felvételéről van szó a program előállítja és meghívja a megfelelő XML formátumú NDA-protokoll kéréseket. Ha a kérés sikeres volt akkor bezáródik az ablak. Hiba esetén kiíródik a hibaüzenet. Ha egyszerű név kiválasztásról van szó, akkor a kiválasztott név és névhordozójának azonosítója átadódik a nyitó ablaknak.

1.63. searchresult.php

Ez a fájl tartalmazza annak a programnak a forráskódját, amely a keresési találati listát megjeleníti. A program a keresési paraméterek alapján előállítja és meghívja a keresést végző XML formátumú NDA-protokoll kérésekt. A válasz szintén XML formátumban érkezik, amelyet egy fa szerkezetű asszociatív tömbbé alakítva dolgoz fel a program az adatok megjelenítéséhez.

1.64. statisztika2.php

Ez a fájl hívja meg az egyes statisztikákat megjelenítő php fájlokat, amelyek a „statisztika” alkönyvtárban találhatóak.

1.65. tipuslistaszerkesztes.php

Ennek a fájlnak a segítségével jeleníthetjük meg és szerkeszthetjük a rendszer működéséhez szükséges típusértékeket.

1.66. webinterface5.php

Ez a fájl jeleníti meg a keresés, illetve a speciális keresés eredményeképpen előállt találati listában szereplő nevek adatlapját.

1.67.webinterface7.php

Ez a fájl jeleníti meg a keresési találati listát a névválasztásnál.

1.68.statisztika könyvtár

1.68.1. stat1.php

Ez a fájl jeleníti meg az *„Összes névtér rekordszám”* statisztikát.

1.68.2. stat2.php

Ez a fájl jeleníti meg és frissíti a *„Névterenkénti rekordszám”* statisztikát.

1.68.3. stat3.php

Ez a fájl jeleníti meg a *„Leggyakoribb névtértípus használat”* statisztikát.

1.68.4. stat4.php

Ez a fájl jeleníti meg és frissíti a *„Legtöbbet referált névtér rekordok”* statisztikát.

1.68.5. stat5.php

Ez a fájl jeleníti meg az *„Új, módosított rekordok”* statisztikát.

1.68.6. stat6.php

Ez a fájl jeleníti meg az *„Összes aktív adatgazda száma”* statisztikát.

1.68.7. stat8.php

Ez a fájl jeleníti meg a *„Legújabb adatgazdák”* statisztikát.

1.68.8. stat9.php

Ez a fájl jeleníti meg és frissíti a *„Legnagyobb adatgazdák”* statisztikát.

1.68.9. stat10.php

Ez a fájl jeleníti meg és frissíti a *„Adott adatgazdához tartozó nevek száma”* statisztikát.

1.68.10. stat11.php

Ez a fájl jeleníti meg a *„Hányszor linkeltek az adott intézményre”* statisztikát.

1.68.11. stat12.php

Ez a fájl jeleníti meg a *„Leggyakoribb keresések, kereső kifejezések”* statisztikát.

1.68.12. stat12a.php

Ez a fájl jeleníti meg a *„Az egyes adatgazdákon belül a leggyakoribb keresések, kereső kifejezések”* statisztikát.

1.68.13. stat13.php

Ez a fájl jeleníti meg a *„Legutóbbi keresések, kereső kifejezések”* statisztikát.

1.68.14. stat14.php

Ez a fájl jeleníti meg a *„Leggyakrabban keresett névtér rekord”* statisztikát.

1.68.15. stat15.php

Ez a fájl jeleníti meg és frissíti a *„Duplikátumok száma”* és a *„Lehetséges duplikátumok száma”*statisztikákat.

1.68.16. stat16.php

Ez a fájl jeleníti meg és frissíti a *„Duplikátumok száma adatgazdánként”* és a *„Lehetséges duplikátumok száma adatgazdánként”* statisztikákat.

1.68.17. stat19.php

Ez a fájl jeleníti meg az *„Összevont rekordok száma”* statisztikát.

1.68.18. stat20.php

Ez a fájl jeleníti meg a *„Összevont rekordok száma adatgazdánként”* statisztikát.

1.68.19. stat21.php

Ez a fájl jeleníti meg a *„Hányan léptek be?”* statisztikát.

1.68.20. stat22.php

Ez a fájl jeleníti meg a *„Hány keresés indult?”* statisztikát.

1.68.21. stat23.php

Ez a fájl jeleníti meg a *„Kattintás szám”* statisztikát.

1.69. param könyvtár

Ebben a könyvtárban az oldal működéséhez szükséges konstans értékeket tartalmazó fájlok találhatók. A könyvtárban található fájlok: params.ini, session.ini, xml.ini

1.70. include könyvtár

Ebben a könyvtárban az XML-kérések értelmetéséhez szükséges konstans értékeket és függvényeket tartalmazó fájlok találhatók. A könyvtárban található fájlok: error_messages.ini, keresertelmezo.inc, keresertelmezo.ini, logs.inc, technames.ini, webinterface.ini, webinterface2.ini, webinterface.inc.

1.71. ke_logs könyvtár

Ebben a könyvtárban a felhasználóknak bejelentkezéskor létrehozott munkamenet fájlok találhatók, amelyek alapján eldönthető, hogy egy adott NDA-protokoll kérés érvényes-e.

1.72. images könyvtár

Ebben a könyvtárban a webes felület megjelenítéséhez szükséges grafikai elemek találhatók.

1.73. uploads könyvtár

Ennek a könyvtárnak az alkönyvtáraiban találhatók az egyes adatgazdák által feltöltött csv fájlok, a visszatérési értékekkel kiegészítve.

1.74. keresertelmezo könyvtár

Ennek a könyvtárban az egyes NDA-protokoll kéréseket végrehajtó segédfájlok vannak.

1.74.1. 01_logink.php

Ez a fájl dolgozza fel a *„Bejelentkezés”* kéréseket. Amennyiben a küldött felhasználónév/jelszó páros érvényes, a függvény előállítja a megfelelő munkafolyamat-azonosítót és visszatér annak az értékével.

1.74.2. 02_search_name.php

Ez a fájl dolgozza fel az *„Általános név keresése megadott karaktersorozat alapján”* kéréseket. A küldött karakterlánc alapján a függvény visszatér az arra a kívánt módon illeszkedő nevek listájával a névtértípustól függetlenül.

1.74.3. 03_special_searchk.php

Ez a fájl dolgozza fel az *„Speciális név keresése megadott karaktersorozat alapján”* kéréseket. A kiválasztott névtértípuson belül, a küldött karakterlánc alapján a függvény visszatér az arra a kívánt módon illeszkedő nevek listájával.

1.74.4. 04_namepart_searchk.php

Ez a fájl dolgozza fel az *„Speciális név keresése megadott névelemre illeszkedő megadott karaktersorozat alapján”* kéréseket. A függvény visszaadja azoknak az elemekre bontott személynevek listáját, melyekben szerepelnek a kérésben meghatározott típusú és értékű névelemek.

1.74.5. 05_extended_searchk.php

Ez a fájl dolgozza fel a *„Speciális név keresése a névhez kapcsolható adatok alapján”* kéréseket.
A függvény visszaadja azoknak a neveknek a listáját, amelyek illeszkednek a kérésben küldött feltételek mindegyikére.

1.74.6. 06_nameholderdatasheet.php

Eza fájl dolgozza fel a *„Névhordozóhoz tartozó adatok megtekintése”* kéréseket. A függvény visszaadja küldött névhordozó azonosítóhoz tartozó adatokat. A visszaadott adatok köre a küldött paramétertől függ.

1.74.7. 07_namedatasheetk.php

Ez a fájl dolgozza fel a *„Névhez tartozó adatok megtekintése”* kéréseket. A függvény visszaadja küldött név azonosítóhoz kapcsolódó adatokat. A visszaadott adatok köre a küldött paramétertől függ.

1.74.8. 13_get_event.php

Ez a fájl dolgozza fel az *„Esemény adatainak megtekintése”* kéréseket. A függvény visszaadja a küldött esemény azonosítóhoz kapcsolódó összes adatot.

1.74.9. 14_get_event.php

Ez a fájl dolgozza fel az *„Eseménynév megtekintése”* kéréseket. A függvény visszaadja a küldött esemény azonosító nevét és annak tulajdonságait.

1.74.10. 16_add_event.php

Ez a fájl dolgozza fel a *„Események felvitele névhez vagy névhordozóhoz”* kéréseket. A függvény a küldött név,- vagy névhordozó azonosítóhoz felveszi a megadott tulajdonságú eseményt.

1.74.11. 17_delete_event.php

Ez a fájl dolgozza fel a *„Esemény kapcsolódás megszüntetése”* kéréseket. A függvény törli a küldött eseményazonosító és a küldött név,- vagy névhordozó azonosító kapcsolatát.

1.74.12. 17_modify_event.php

Ez a fájl dolgozza fel a *„Névhordozó vagy név kapcsolása eseményhez”* kéréseket. A függvény összekacsolja a küldött név,- vagy névhordozó azonosítót a küldött eseményazonosítóval.

1.74.13. 19_edit_event.php

Ez a fájl dolgozza fel a *„Esemény szerkesztése”* kéréseket. A függvény módosítja a küldött eseményazonosítóhoz tartozó időértékeket.

1.74.14. 21_add_remove_related_object.php

Ez a fájl dolgozza fel a *„Objektumok közötti kapcsolódás létrehozása/megszüntetése”* kéréseket. Ha a küldött objektumtípus, kapcsolat, objektumtípus hármas érvényes és akkor a függvény létrehozza a kapcsolatot, ha már létezett ez a kapcsolat, akkor törli.

1.74.15. 23_edit_object.php

Ez a fájl dolgozza fel a *„Objektumok tulajdonságainak szerkesztése”* kéréseket. A függvény módosítja a küldött névhordozó tulajdonságait a küldött értékekkel.

1.74.16. 24_modify_name.php

Ez a fájl dolgozza fel a „Névhez tartozó adatok szerkesztése” kéréseket. A függvény módosítja a küldött név tulajdonságait a küldött értékekkel.

1.74.17. 25_add_object.php

Ez a fájl dolgozza fel a *„Névhordozó felvitele”* kéréseket. A függvény felvisz a névtérbe egy új névhordozót a megadott tulajdonságokkal.

1.74.18. 26_add_name.php

Ez a fájl dolgozza fel a *„Név felvitele”* kéréseket. A függvény felvisz a névtérbe egy új nevet a megadott tulajdonságokkal.

1.74.19. 28_assign_name.php

Ez a fájl dolgozza fel a *„Név kapcsolása névhordozóhoz”* kéréseket. A függvény a küldött névhordozóhoz kapcsolja a küldött nevet. Abban az esetben amikor a névhordozónak már létezik olyan neve, aminek a névalakja megegyezik a hozzá kapcsolni kívánt név névalakjával, akkor az új név minden tulajdonságát felvisszük a névhordozóhoz kapcsolódó névhez.

1.74.20. 29_add_note.php

Ez a fájl dolgozza fel a *„Megjegyzés felvitele névhez vagy névhordozóhoz”* kéréseket. A függvény a megadott tulajdonságokal rendelkező megjegyzést fűzi a küldött név,- vagy névhordozó azonosítóhoz.

1.74.21. 30_edit_note.php

Ez a fájl dolgozza fel a *„Megjegyzés szerkesztése”* kéréseket. A függvény módosítja a küldött megjegyzésazonosítóhoz tartozó adatokat.

1.74.22. 32_edit_footprint.php

Ez a fájl dolgozza fel a *„Földrajzi objektum footprint adatainak szerkesztése”* kéréseket. A függvény módosítja a küldött földrajzi objektumhoz tartozó footprint adatokat.

1.74.23. 33_edit_name_parts.php

Ez a fájl dolgozza fel a *„Névelemekre bontás szerkesztése”* kéréseket. A függvény módosítja a küldött személy típusú objektumhoz tartozó névelemekre bontást.

1.74.24. 34_get_owner.php

Ez a fájl dolgozza fel a *„Adatgazda adatlapjának megtekintése”* kéréseket. A függvény visszaadja a küldött adatgazda azonosítóhoz tartozó adatokat.

1.74.25. 35_list_modrequests.php

Ez a fájl dolgozza fel a *„Módosítási kérések listázása”* kéréseket. A függvény visszadja a módosítási kérelmek listáját.

1.74.26. 36_get_modrequests.php

Ez a fájl dolgozza fel a *„Módosítási kérés megtekintése”* kéréseket. A függvény visszadja a küldött módosítási kérelem azonosítóhoz tartozó módosítási kérést.

1.74.27. 37_save_mod.php

Ez a fájl dolgozza fel a *„Módosítási kérés elfogadása”* kéréseket. A függvény végrehajtja a küldött módosítási kérelem azonosítóhoz tartozó kérést.

1.74.28. 38_list_states.php

Ez a fájl dolgozza fel a *„Név vagy névhordozó történetének megtekintése”* kéréseket. A függvény kilistázza a küldött név,- vagy névhordozó azonosítóhoz tartozó módosításokat.

1.74.29. 39_get_state.php

Ez a fájl dolgozza fel a *„Név vagy névhordozó múltbeli állapotának megtekintése”* kéréseket. A függvény visszaadja a küldött múltbeli állapot azonosító alaján a név, vagy névhordozó múltbeli állapotát.

1.74.30. 40_list_objecttypes.php

Ez a fájl dolgozza fel a *„Névtértípusok lekérése”* kéréseket. A függvény a névtértípusok azonosítóját és nevét tartalmazó listát adja vissza.

1.74.31. 41_otherside_types.php

Ez a fájl dolgozza fel a *„Lehetséges névtértípusok lekérése egy kapcsolathoz”* kéréseket. A függvény visszadja a küldött kapcsolattípus azonosítóhoz és névtértípus azonosítóhoz tartozó jobb,- vagy baloldalon kapcsolódó névtértípusokat.

1.74.32. 42_otherside_types.php

Ez a fájl dolgozza fel a *„Lehetséges névtértípusok lekérése egy kapcsolathoz”* kéréseket. A függvény visszadja a küldött kapcsolattípus azonosítóhoz tartozó névtértípusokat.

1.74.33. 43_is_valid_types.php

Ez a fájl dolgozza fel a *„Névtértípus, kapcsolat, névtértípus hármas sorrendjének ellenőrzése”* kéréseket. A függvény megvizsgálja, hogy a küldött névtértípus, kapcsolat, névtértípus hármas érvényes-e.

1.74.34. 44_list_types.php

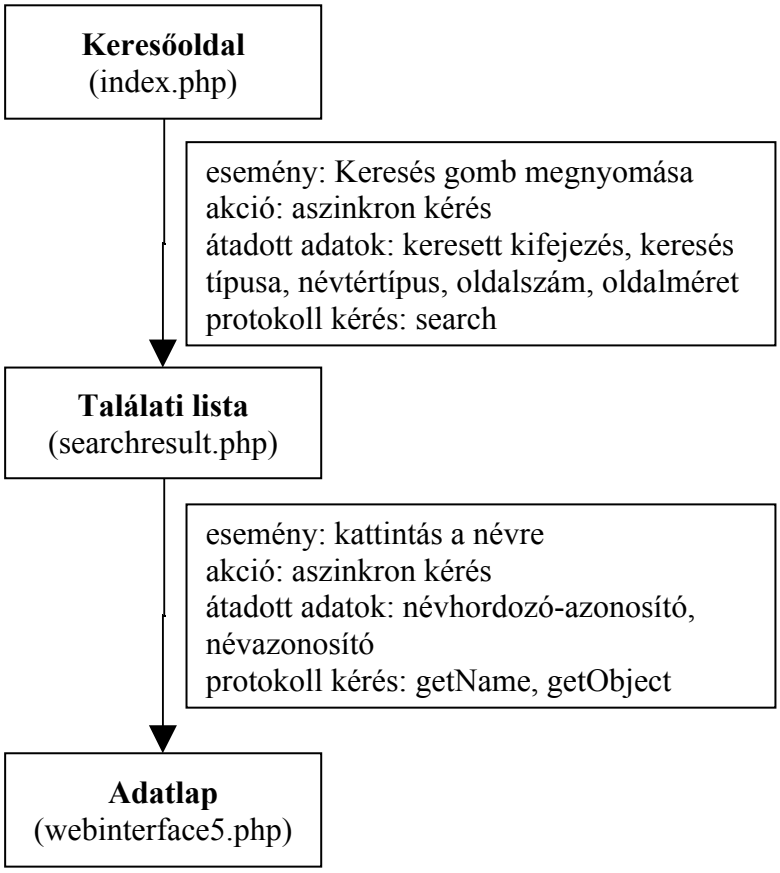
Ez a fájl dolgozza fel a *„Kategória alá tartozó típusok lekérése”* kéréseket. A függvény visszaadja a küldött kategória azonosítóhoz tartozó típusértékeket.

1.74.35. 45_list_root_types.php

Ez a fájl dolgozza fel a *„Őstípusok lekérése”* kéréseket. A függvény visszadja azoknak a típusoknak a listáját, amelyeknek nem tartoznak egyetlen típushoz sem.

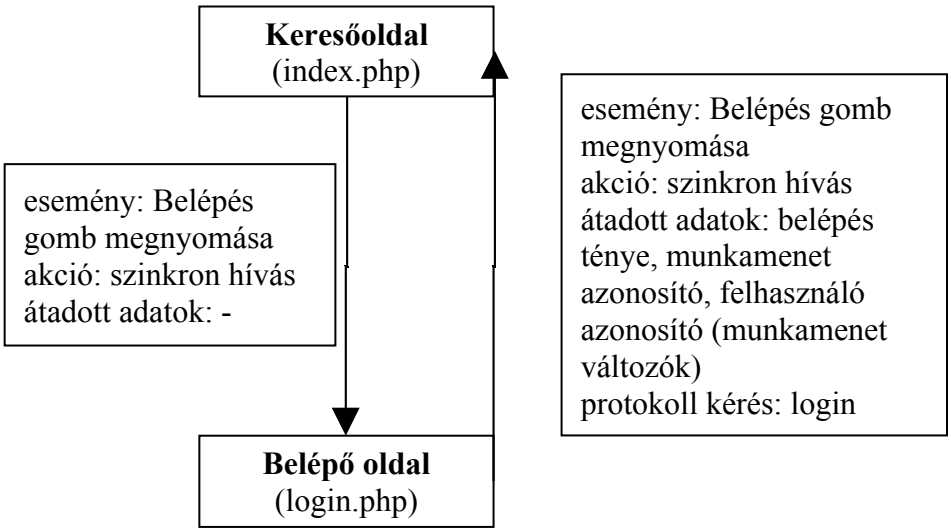
2. A forrásfájlok közötti kapcsolatok szemléletesen

2.1. Keresés, adatlap megtekintése



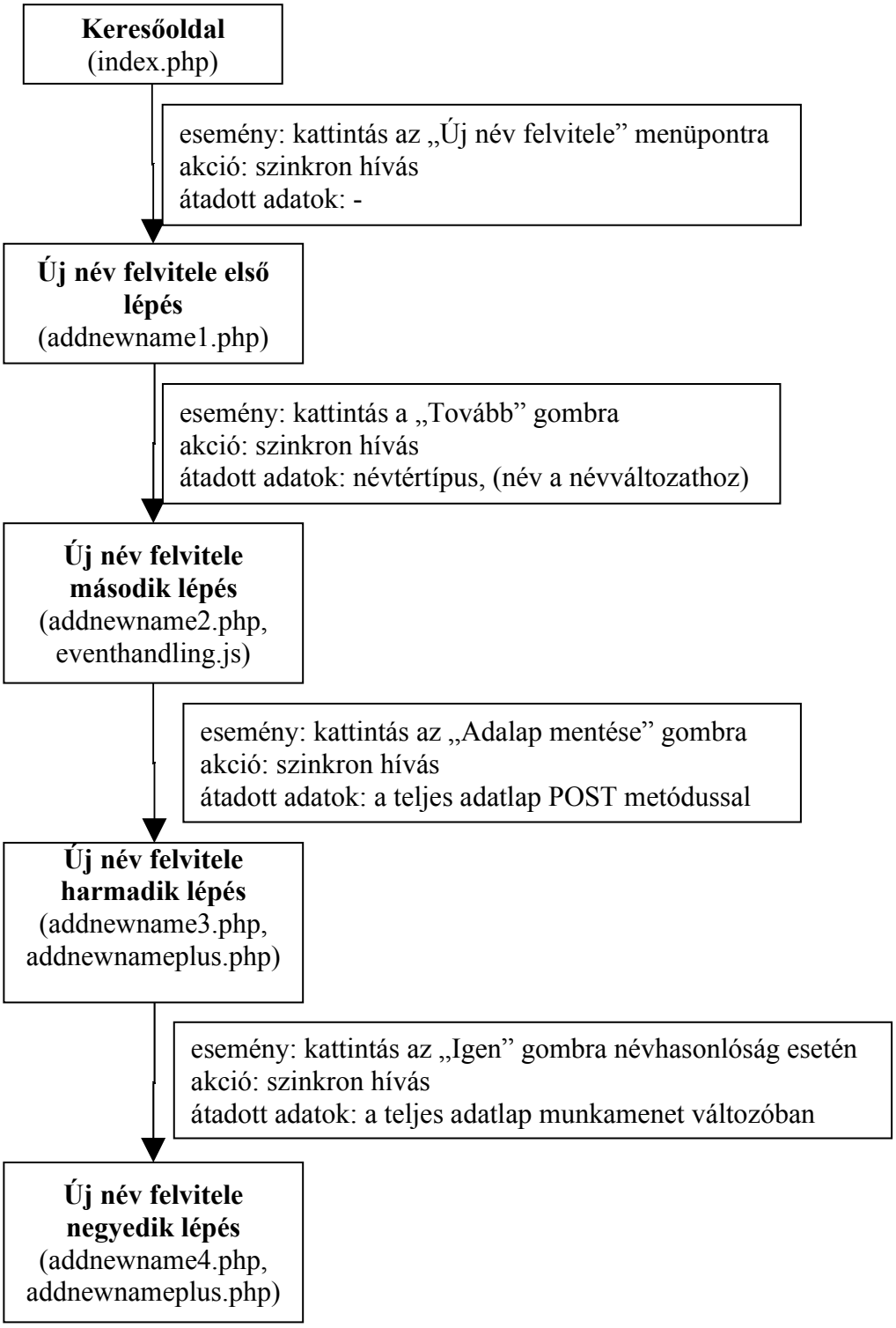
Az index.php fájl állítja össze a kereső űrlapot, amelyen megadhatjuk, hogy milyen szövegre, szóeleji, vegy szóközepi egyezéssel, milyen névtérben keressünk, és oldalanként hány nevet megjelenítve listázzuk ki a találatokat. A „Keresés” gombra kattintva meghívódik a searchNames javascript függvény, amely a keresési paraméterek alapján összeállítja azt az aszinkron http kérést, amire válaszként a találati lista HTML kódja érkezik. A searchNames javascript függvény, a keresési paramétereket POST metódussal küldi el és a searchresult.php fájlt hívja meg. A searchresult.php a $_POST változóban érkező keresési paraméterek alapján összeállítja a keresést végző, XML szerkezetű NDA-protokoll kérést, a kérés szintén XML szerkezetű válasza alapján pedig összeállítja a találati lista HTML kódját.
A találati listában egy névre kattintva meghívódik a showDatasheet javascript függvény, amely összeállítja azt az aszinkron http kérést, amire válaszként a névhez tartozó adatok HTML kódja érkezik. Az adatlap lekéréséhez szükséges névhordozó-azonosítót és névazonosítót a függvény POST metódussal küldi el, és a webinterface5.php fájlt hívja meg. A webinterface5.php a $_POST változóban érkező adatok alapján összeállítja a névhez tartozó adatokat lekérő, XML szerkezetű NDA-protokoll kéréseket, a kérések szintén XML szerkezetű válaszai alapján pedig összeállítja az adatlap HTML kódját.
A résletes keresés hasonlóan működik, de ott a keresési űrlapot az extendedSearch.php jeleníti meg és a feltételek alapján összeállított keresési kérést is ez a fájl állítja össze.

2.2. Bejelentkezés



A „Belépés” gombra kattintva betöltődik a login.php oldal, ahol a regisztrált felhasználó (névtér adatgazda) megadhatja a felhasználónevet és a jelszót. A fájl ekkor újratöltődik és a bejelentkezési adatokat POST metódussal elküldi a szervernek. Az újratöltés után a $_POST változóban érkező felhasználónév és jelszó alapján a program összeállítja az XML szerkezetű login NDA-protokoll kérést, a kérés szintén XML szerkezetű válasza alapján pedig a PHP munkamenet változóban eltárolja a bejelentkezés tényét, a munkamenet azonosítót és a felhasználó azonosítót. Sikeres bejelentkezés esetén a program betölti a keresőoldal, sikertelen bejelentkezésnél újra a bejelentkező űrlap jelenik meg.

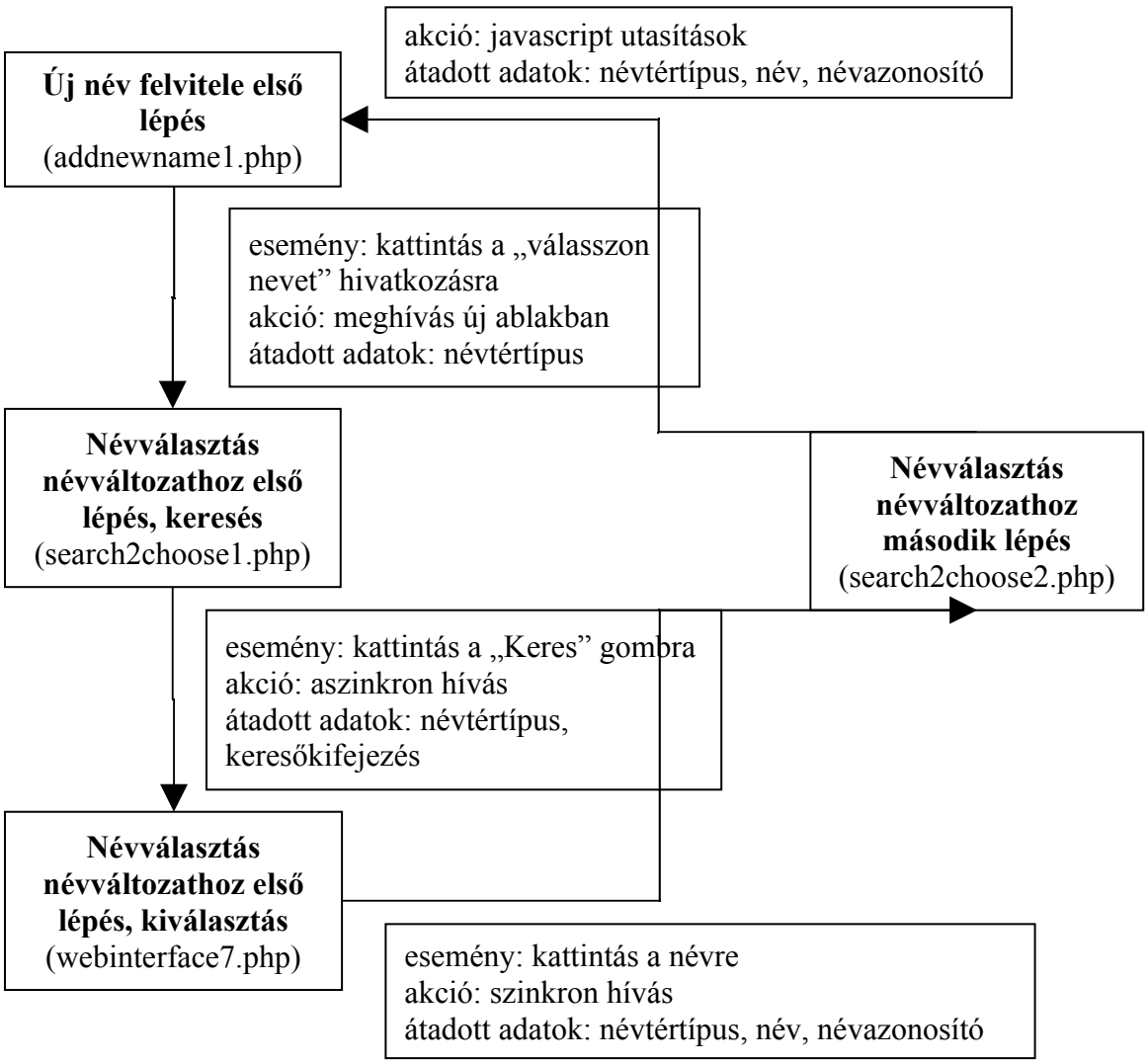
2.3. Új név felvitele



Az új név felvitelének első lépését az addnewname1.php végzi. Itt kell kiválasztani a névtértípust, illetve ha névváltozatot akarunk felvinn, akkor azt, hogy melyik névhez tesszük ezt. A „Tovább” gombra kattintva betöltődik az addnewname2.php fájl amely a $_POST változóban kapja meg, hogy milyen típusú nevet akarunk felvenni, illetve új névváltozat esetén a korábbi névhez tartozó név és névhordozó azonosítót. Az addnewname2.php fájl

állítja össze és működteti az új név adatainak megadására szolgáló űrlapot. Névváltozat felvitele esetén ez a program jeleníti meg a névhordozó adatait. Az „Adalap mentése” gombra kattintva megyünk tovább az új név felvitelének harmadik lépésére, az addnewname3.php fájlra. A név adatai POST metódussal adódnak át. Az addnewname3.php fájl végzi az új név mentését. Ha az új névhez már létezett hasonló a rendszerben, akkor erre a program figyelmeztet és feltesz egy megerősítő kérdést a név felvételére vonatkozóan. Ha a válasz igen, akkor következik az új név felvitelének negyedik lépése és betöltődik az addnewname4.php fájl. Az addnewname3.php és az addnewname4.php is meghívja az addnewnameplus.php fájlt, amely az események, eseményen keresztül kapcsolódó nevek, kapcsolódó objektumok, megjegyzések, személyeknél a névelemek, földrajzi objektumoknál a geometriai adatok mentését végzi.

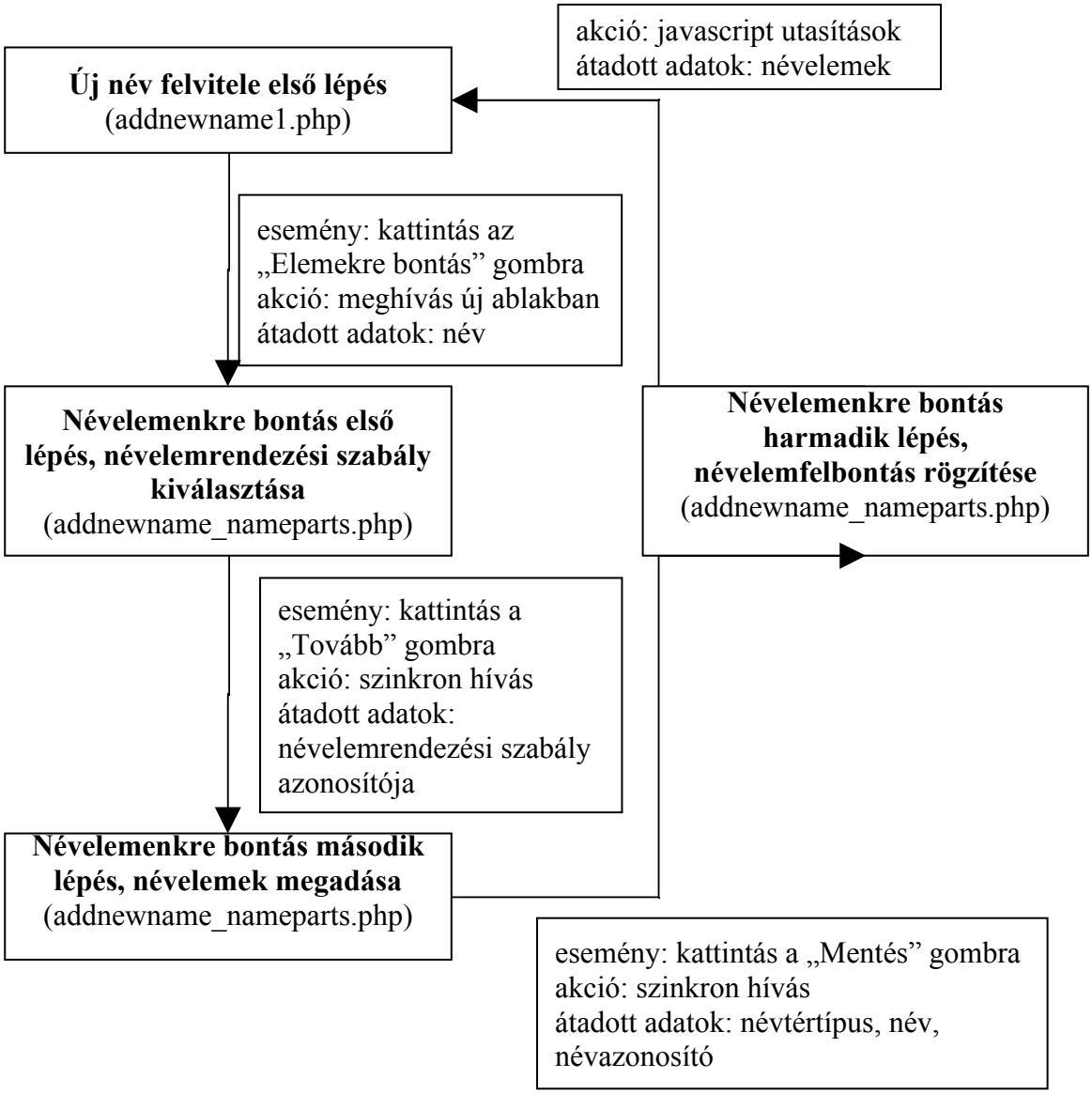
2.3.1. Névválasztás névváltozathoz



Ha egy meglévő névhez szeretnénk új névváltozatot felvenni, akkor az új név felvitelének első lépésénél a „válasszon nevet” hivatkozásra kattintva tudunk nevet választani. Az új abalakban megjelenő kerső űrlap segítségével jeleníthetjük meg a kiválasztandó nevet. Itt a keresési találatokat a webinterface7.php fájl jeleníti meg. A névre kattintva meghívódik a

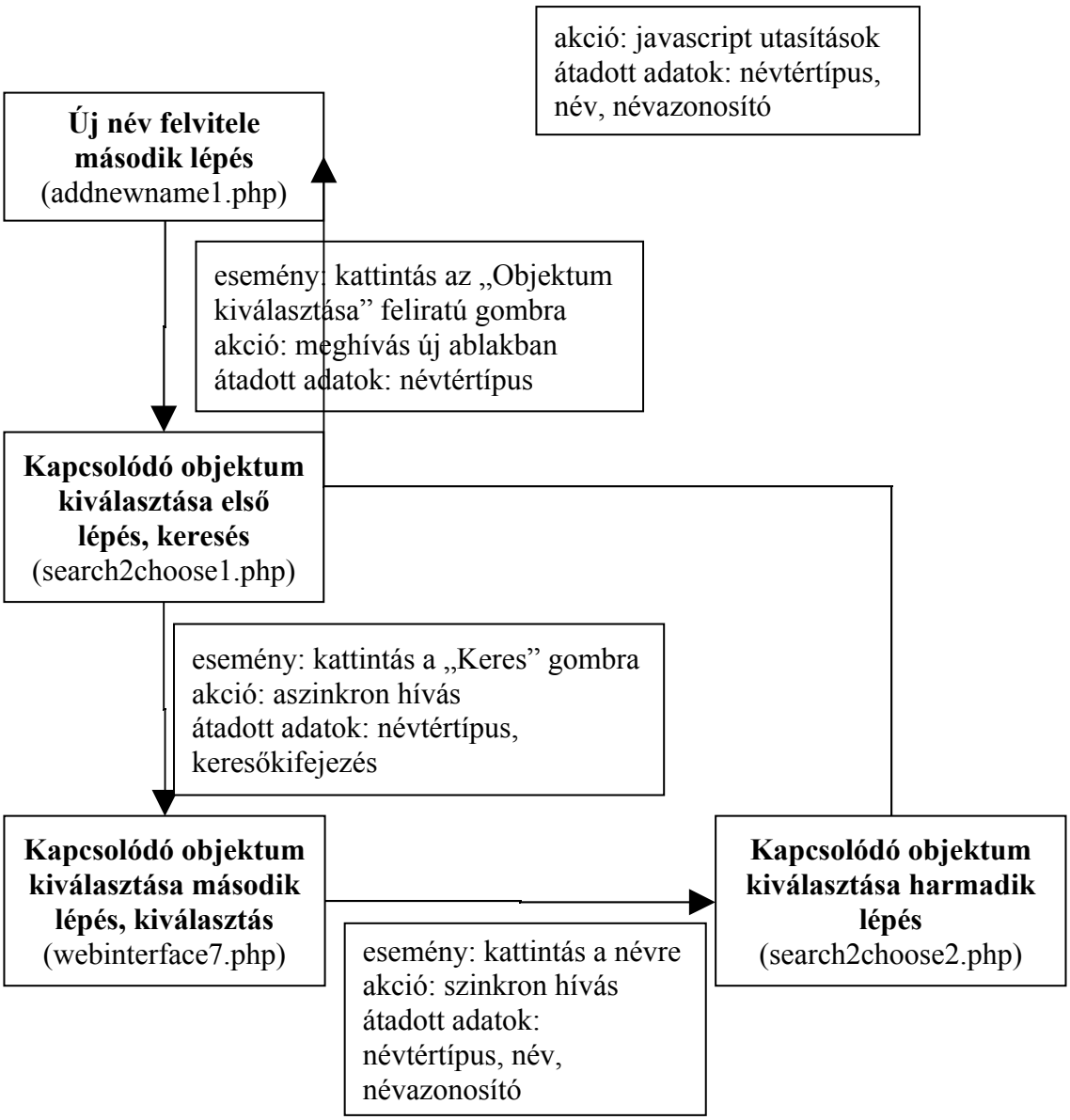
search2choose2.php fájl, amely javaScript utasítások segítségével adja át a kiválasztott név adatait az új név felvitelének első lépéséhez.

2.3.2. Névelemszerkesztés



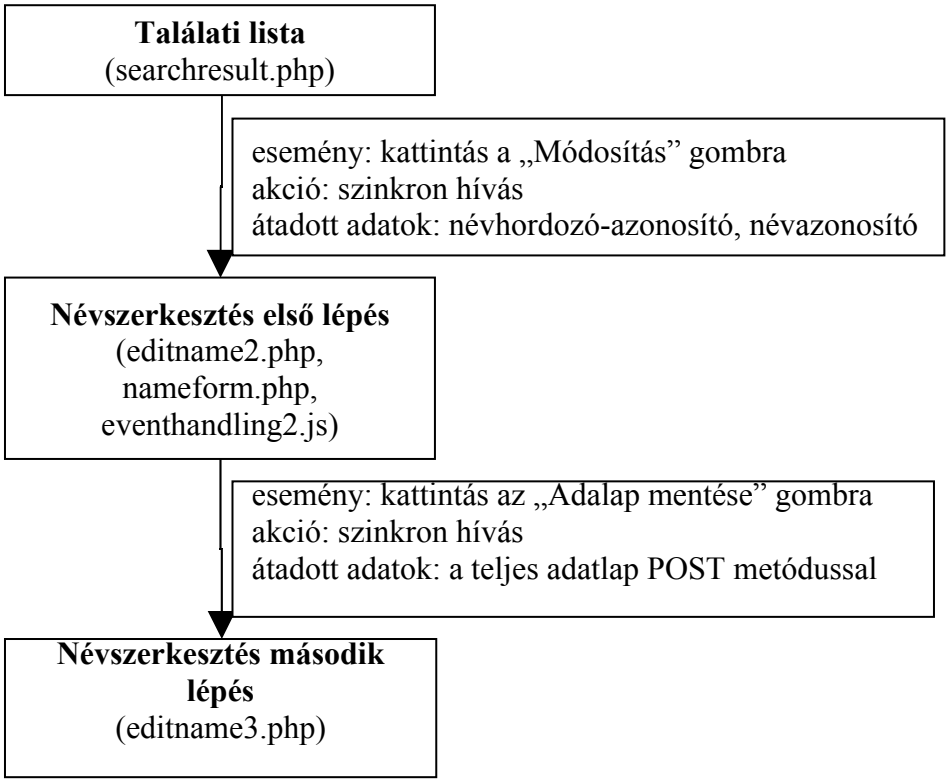
Új név név megadásánál az „Elemekre bontás” feliratú gombra kattintva új ablakban meghívódik az addnewname_nameparts.php fájl, amely a név elemenkénti megadását működteti. Az első lépésben a névelemrendezési szabályt kell kiválasztani. A második lépésben a névelemeket kell megadni, majd a „Mentés” gomb megnyomása után a folyamat harmadik lépéseként a névelemenként megadott név javscript utasításokkal átadódik a nyitó ablaknak.

2.3.3. Eseményen keresztül kapcsolódó név, kapcsolódó objektum kiválasztása



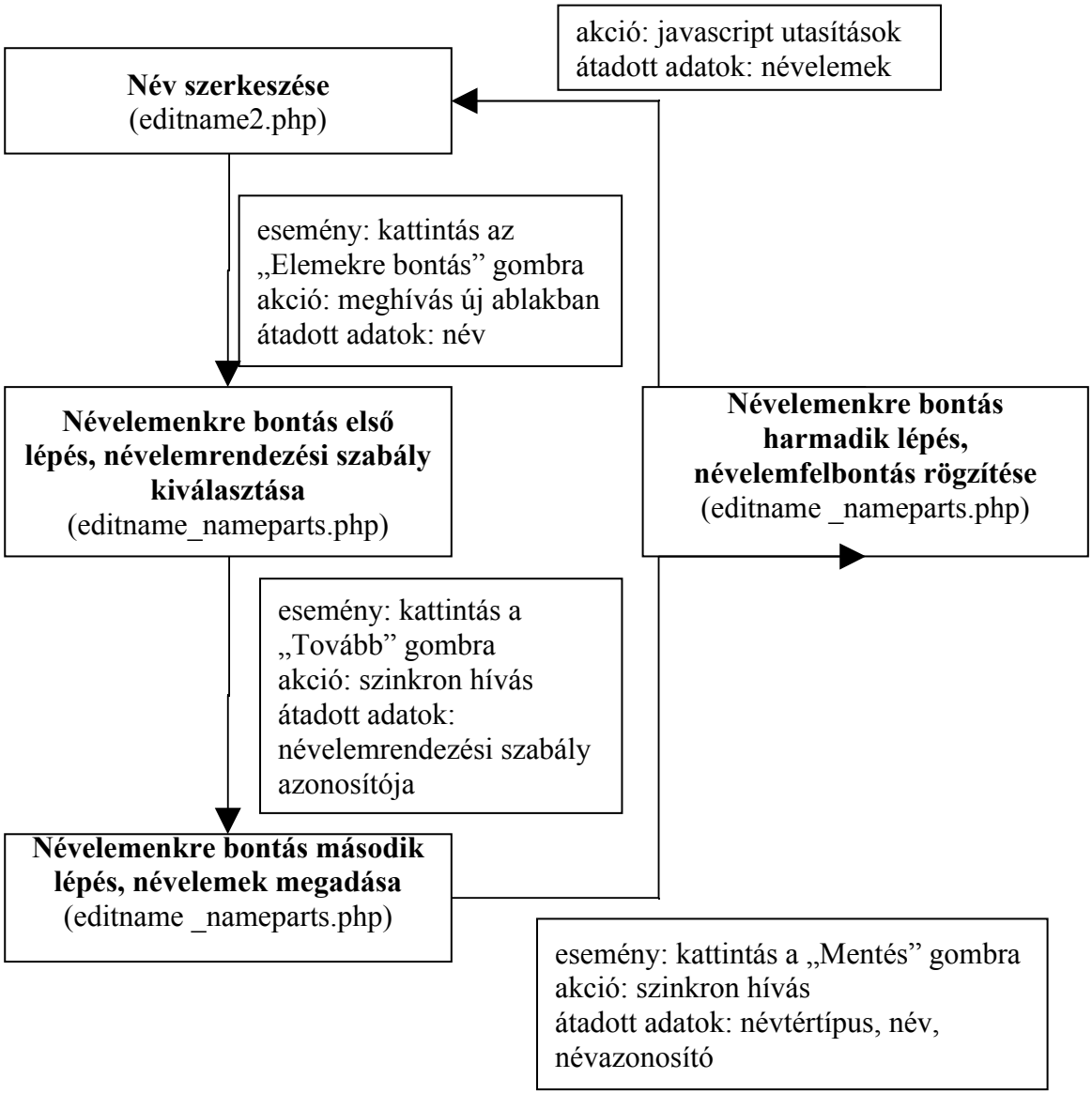
Ha eseményen keresztül kapcsolódó nevet, vagy kapcsolódó objektumot szeretnénk felvenni, az új név felvitelének második lépésénél, akkor az „Objektum kiválasztása” feliratú gombra kattintva tudunk keresni. Az új abalakban megjelenő kerső űrlap segítségével jeleníthetjük meg a kiválasztandó nevet. Itt a keresési találatokat a webinterface7.php fájl jeleníti meg. A névre kattintva meghívódik a search2choose2.php fájl, amely javaScript utasítások segítségével adja át a kiválasztott név adatait az új név felvitelének első lépéséhez.

2.4. Adatlap szerkesztése



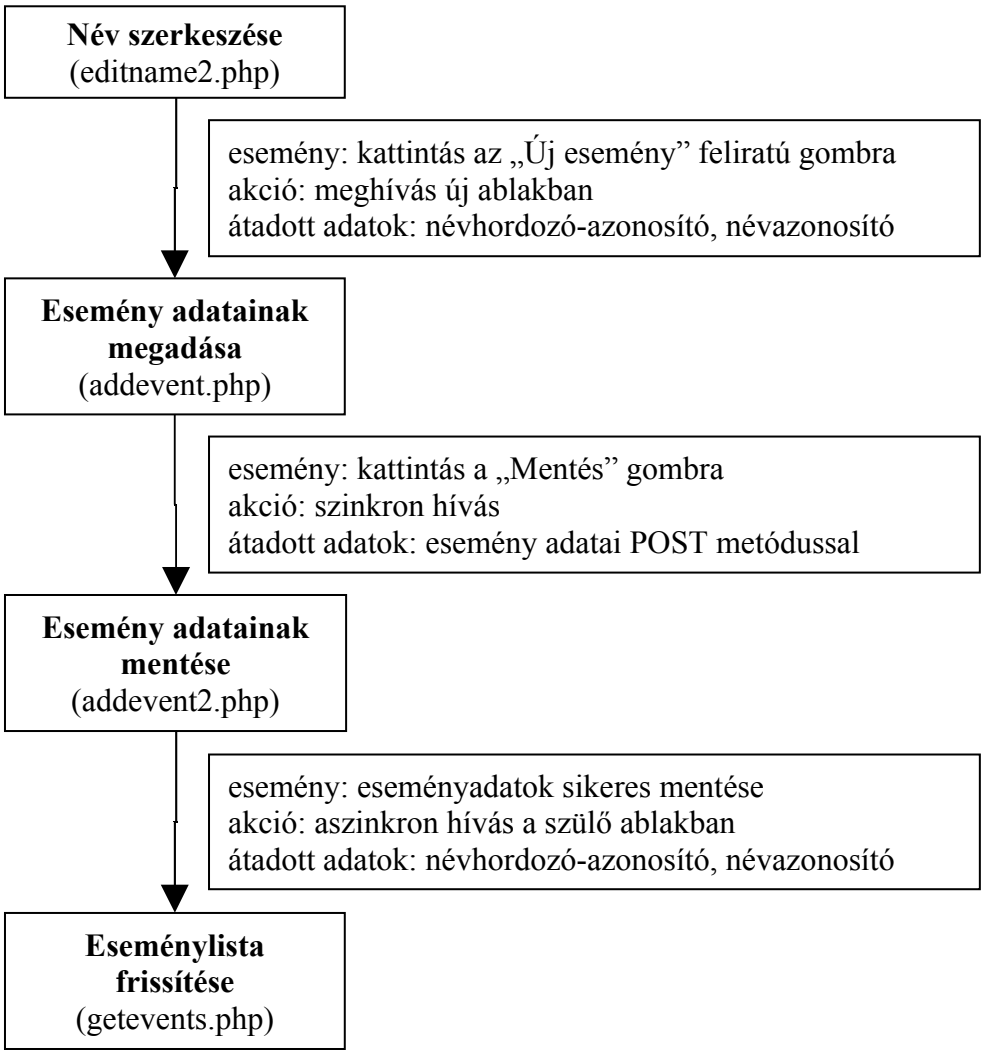
A nevek adatait csak regisztrált névtér adatgazda tudja szerkeszteni, azután, hogy sikeresen bejelentkezett. Ebben az esetben a keresési találati listában megjelenő „Módosítás” feliratra kattintva érhető el a nevek adatainak szerkesztésére szolgáló űrlap. Ennek forráskódját az editname2.php és a nameform.php fájlok tartalmazzák.. A szerkesztő űrlapon megadott adatokat az „Adalap mentése” gombra kattintva lehet véglegesíteni. Ekkor meghívódik az editname3.php fájl, amely az adatok mentését végző XML formátumú NDA-protokoll kéréseket legenrálja, és a válasz alapján kiírja a mentés eredményességét jelző üzenetet.

2.4.1. Névelemszerkesztés



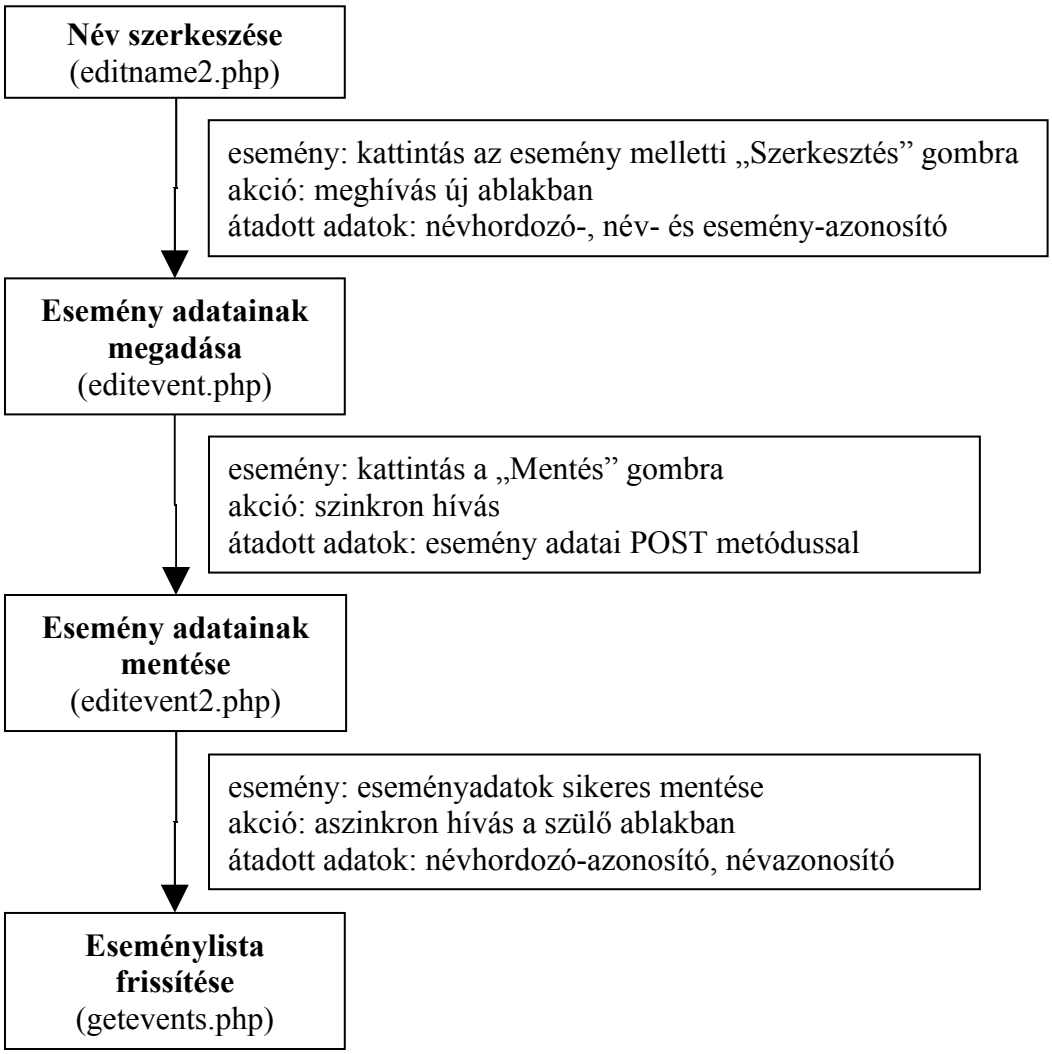
Névszerkesztésnél az „Elemekre bontás” feliratú gombra kattintva új ablakban meghívódik az editname_nameparts.php fájl, amely a név elemenkénti megadását működteti. Az első lépésben a névelemrendezési szabályt kell kiválasztani. A második lépésben a névelemeket kell megadni, majd a „Mentés” gomb megnyomása után a folyamat harmadik lépéseként a névelemenként megadott név javscript utasításokkal átadódik a nyitó ablaknak.

2.4.2. Esemény felvitele szerkesztésnél



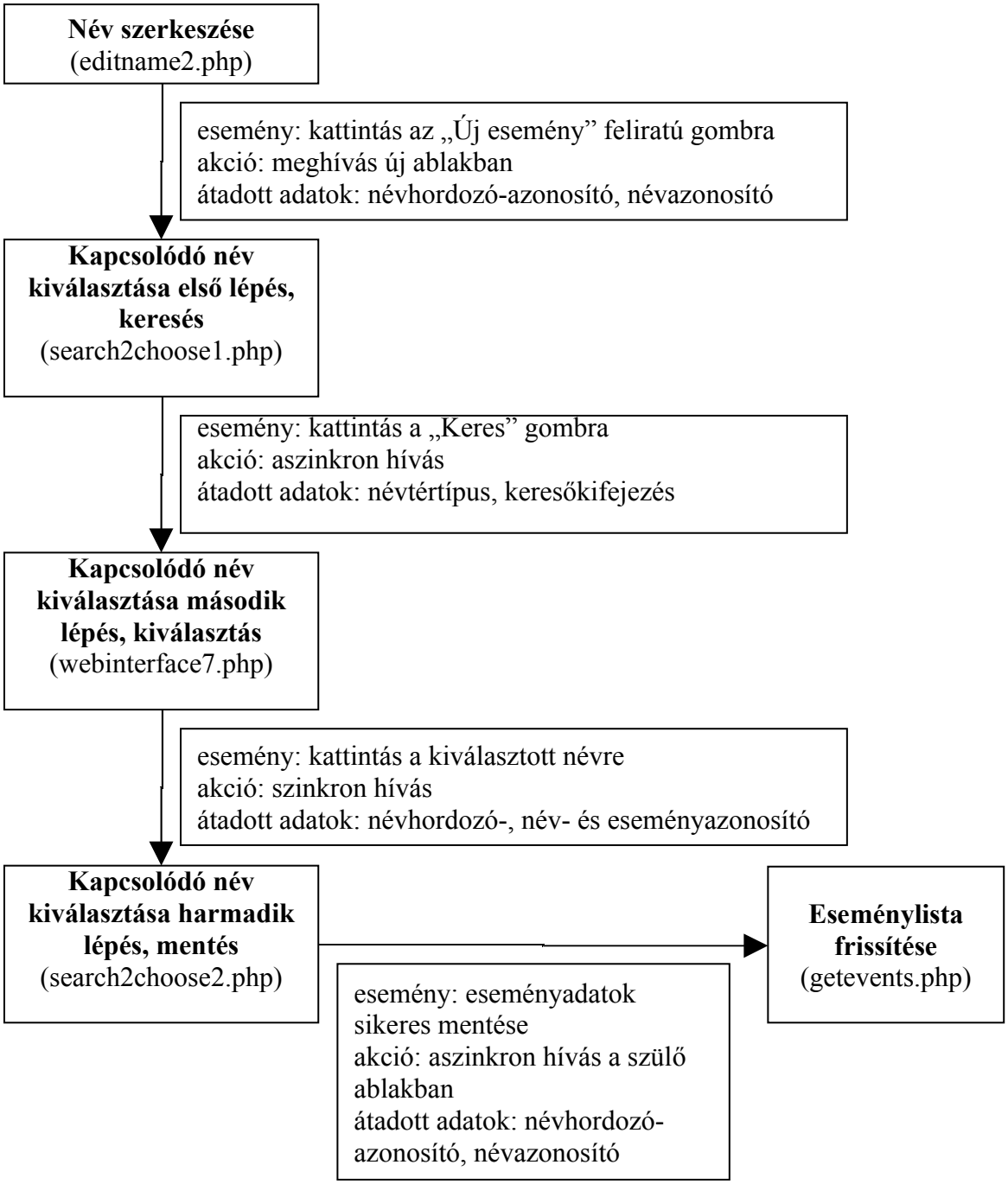
Ha a névszerkesztésnél új eseményt szeretnénk felvenni, akkor az „Új esemény” feliratú gombra kattintva új ablakban megjelenik az esemény adatainak megadására szolgáló űralp, melynek forráskódját az addevent.php fájl tartalmazza. A „Mentés” gomb megnyomására meghívódik az addevent2.php fájl mely az eseményadatok mentését végzi. Az eseményadatok POST metódussal érkeznek, ami alapján a program összeállítja az XML formátumú NDA-protokoll kérést. Ha a mentés sikeres volt, akkor a szerkesztő ablakban frissül az események listája, amelyet a getevents.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.3. Esemény módosítása szerkesztésnél



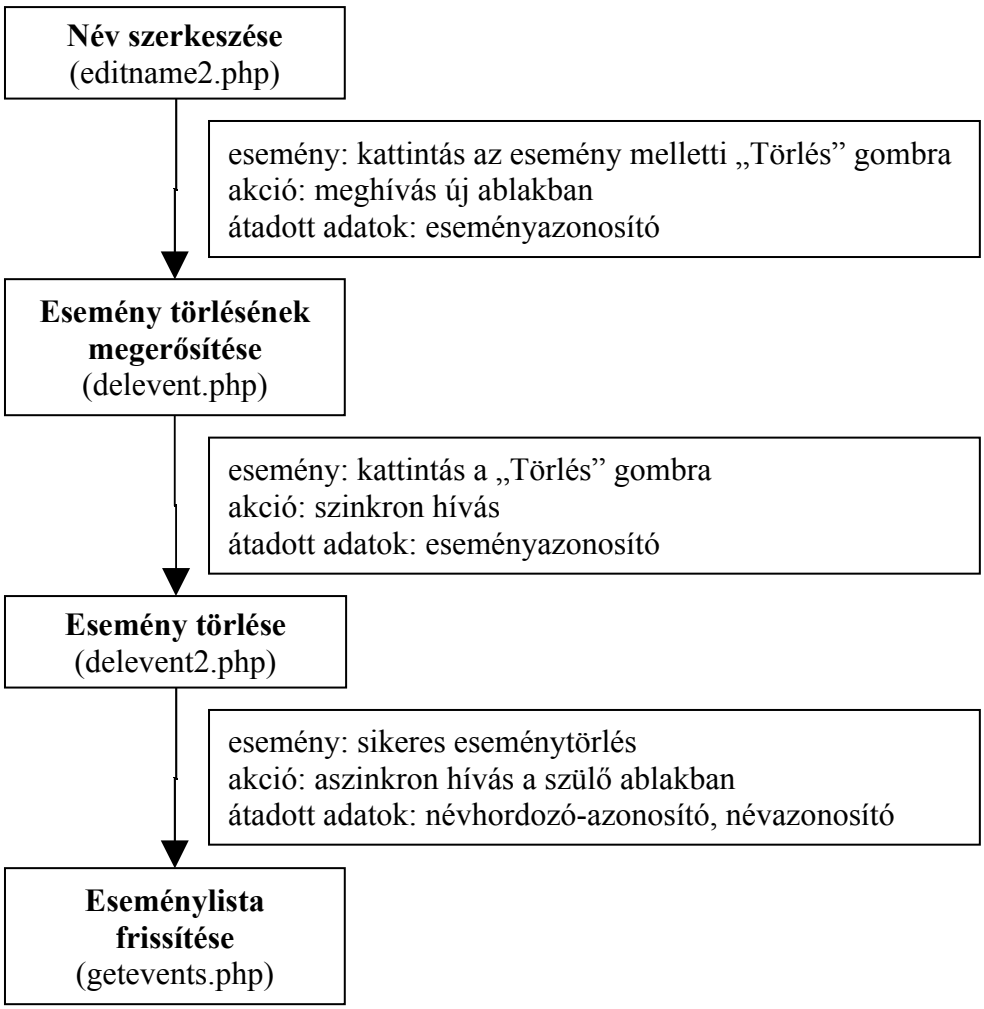
Ha a névszerkesztésnél egy korábban felvitt esemény adatait szeretnénk módosítani, akkor az esemény melletti „Szerkesztés” gombra kattintva új ablakban megjelenik az esemény adatainak módosítására szolgáló űralp, melynek forráskódját az editevent.php fájl tartalmazza. A „Mentés” gomb megnyomására meghívódik az editevent2.php fájl mely az eseményadatok mentését végzi. Az eseményadatok POST metódussal érkeznek, ami alapján a program összeállítja az XML formátumú NDA-protokoll kérést. Ha a mentés sikeres volt, akkor a szerkesztő ablakban frissül az események listája, amelyet a getevents.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.4. Eseményen keresztül kapcsolódó név hozzáadása szerkesztésnél



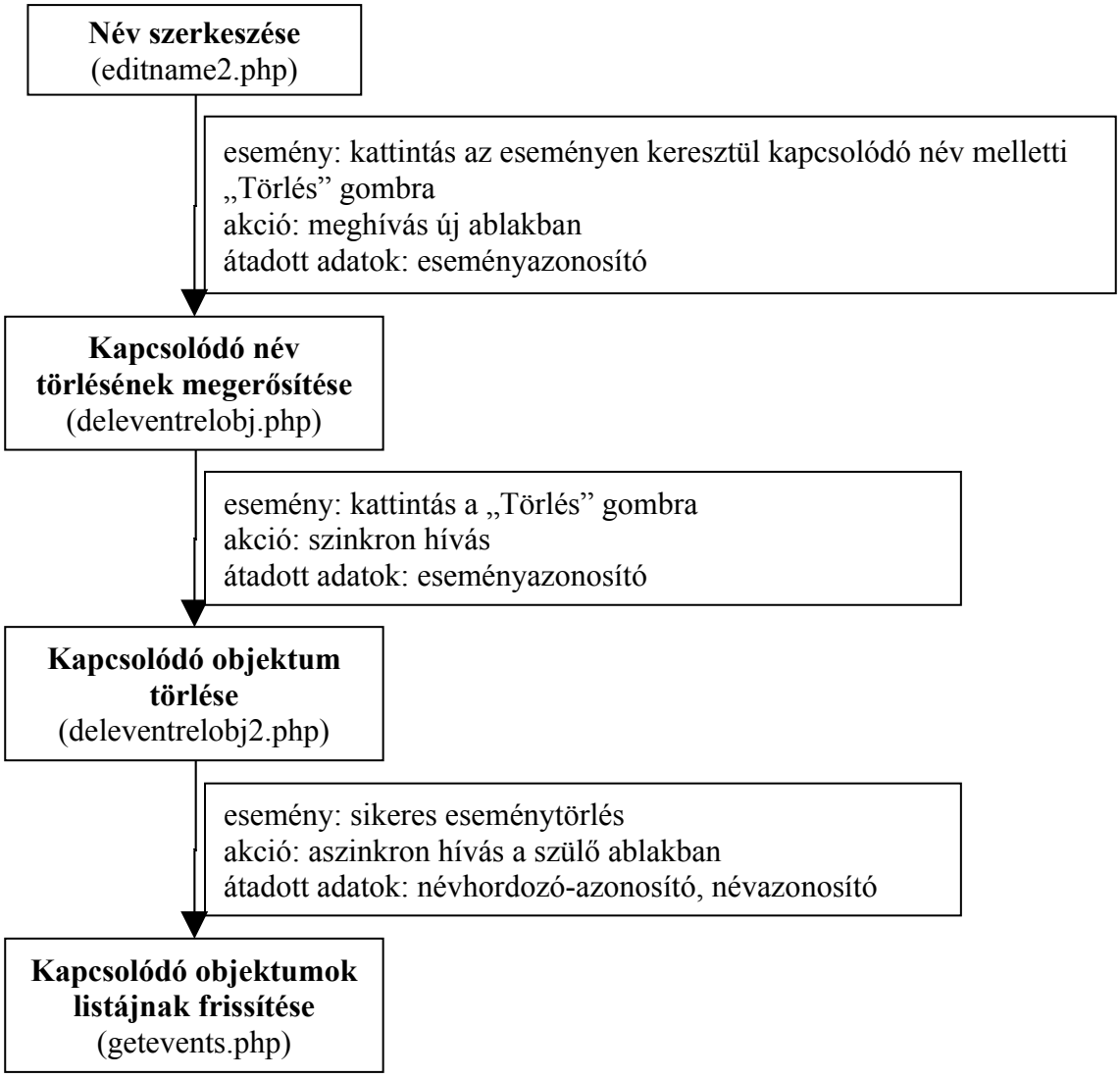
Ha a névszerkesztésnél eseményhez kapcsolódó nevet szeretnénk felvenni, akkor az „Új kapcsolódó név” feliratú gombra kattintva tudunk keresni. Az új abalakban megmegnyíló kerső űrlap segítségével jeleníthetjük meg a kiválasztandó nevet. Itt a keresési találatokat a webinterface7.php fájl jeleníti meg. A névre kattintva meghívódik a search2choose2.php fájl, amely öszzeállítja azt az XML formátumú NDA-protokoll kérést, amely az eseményhez kapcsolódó név mentését végzi. Ha a mentés sikeres volt, akkor a szerkesztő ablakban frissül az események listája, amelyet a getevents.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.5. Esemény törlése szerkesztésnél



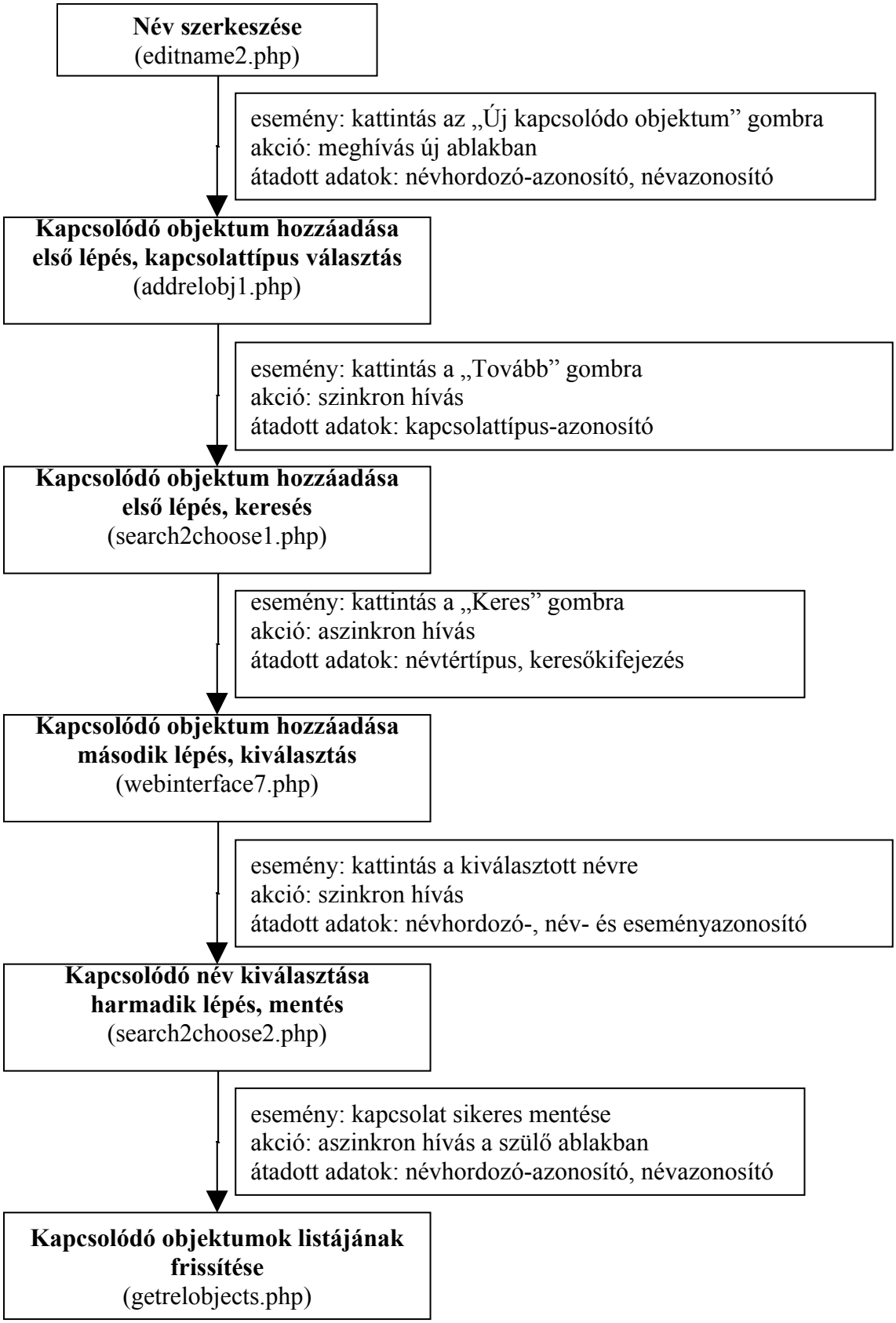
Ha a névszerkesztésnél eseményt szeretnénk törölni, akkor az esemény melletti „Törlés” feliratú gombra kattintva új ablakban megjelenik az esemény törlésének megerősítésére szolgáló kérdés, melynek forráskódját a delevent.php fájl tartalmazza. A „Törlés” gomb megnyomására meghívódik a delevent2.php fájl mely az esemény törlését végzi. A törlendő esemény azonosítója POST metódussal érkezik, ami alapján a program összeállítja az XML formátumú NDA-protokoll kérést. Ha a törlés sikeres volt, akkor a szerkesztő ablakban frissül az események listája, amelyet a getevents.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.6. Eseményen keresztül kapcsolódó név törlése szerkesztésnél



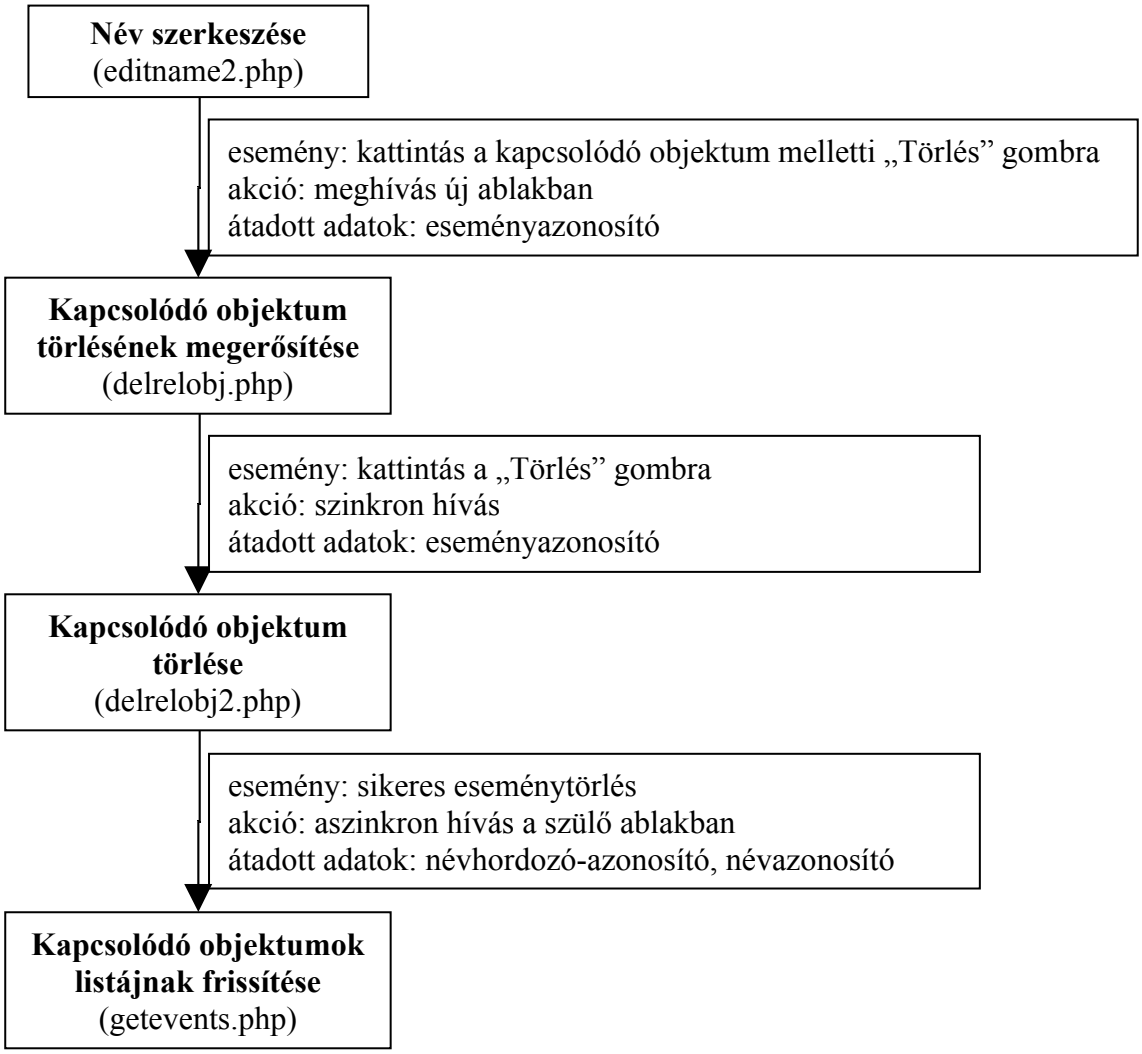
Ha a névszerkesztésnél eseményen keresztül kapcsolódó nevet szeretnénk törölni, akkor a kapcsolódó név melletti „Törlés” feliratú gombra kattintva új ablakban megjelenik a kapcsolódó név törlésének megerősítésére szolgáló kérdés, melynek forráskódját a deleventrelobj.php fájl tartalmazza. A „Törlés” gomb megnyomására meghívódik a deleventrelobj2.php fájl mely az eseményen keresztül kapcsolódó név törlését végzi. A törlendő kapcsolódó név azonosítója POST metódussal érkezik, ami alapján a program összeállítja az XML formátumú NDA-protokoll kérést. Ha a törlés sikeres volt, akkor a szerkesztő ablakban frissül a kapcsolódó objektumok listája, amelyet a getrelobjects.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.7. Kapcsolódó objektum hozzáadása szerkesztésnél



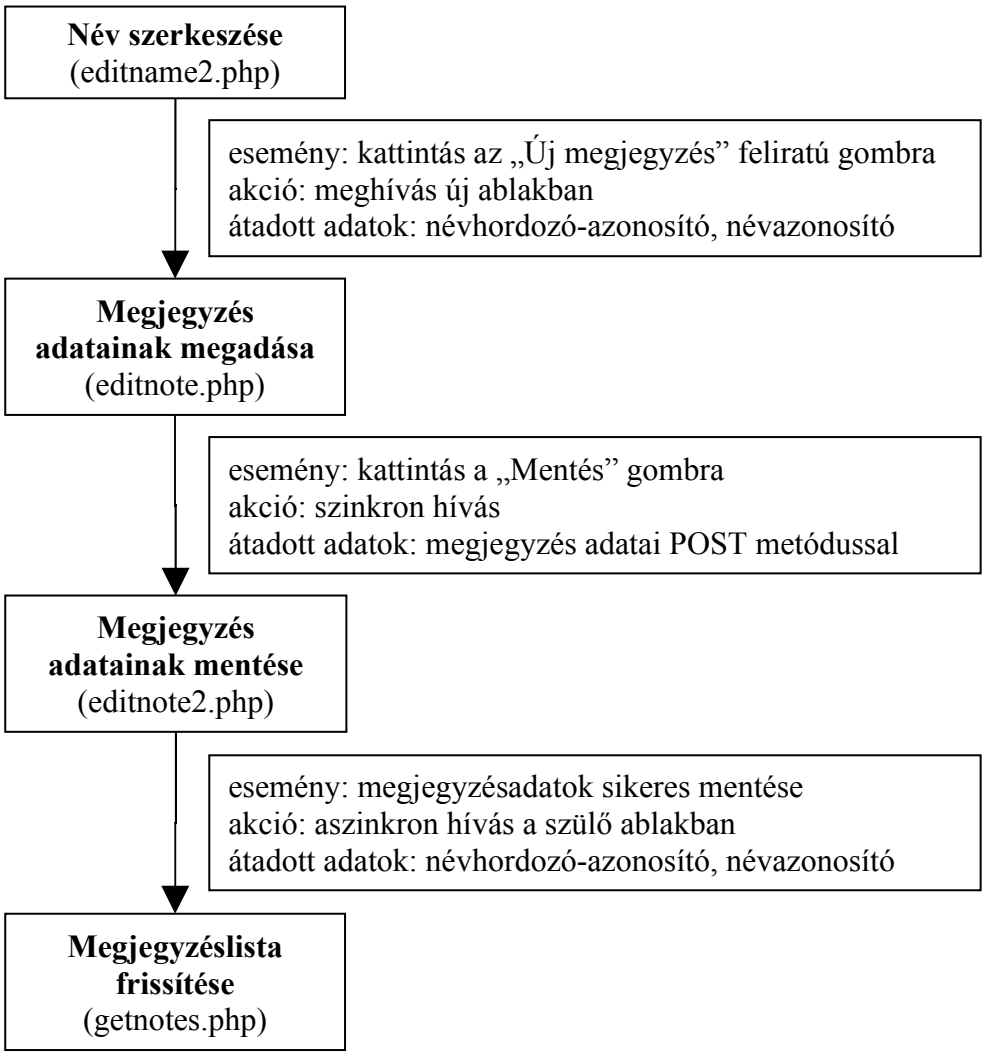
Ha a névszerkesztésnél kapcsolódó objektumot szeretnénk felvenni, akkor az „Új kapcsolódó objektum” feliratú gombra kattintva új ablakban meghívódik az addrelobj1.php fájl, amellyel a kapcsolattípust lehet kiválasztani. A tovább gomb megnyomása után a keresés és a kiválasztás ugyanúgy történik, mint az eseményen keresztül kapcsolódó név kiválasztásánál. A névre kattintva meghívódik a search2choose2.php fájl, amely összeállítja azt az XML formátumú NDA-protokoll kérést, amely a kapcsolódó objektum mentését végzi. Ha a mentés sikeres volt, akkor a szerkesztő ablakban frissül az események listája, amelyet a getrelobjects.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.8. Kapcsolódó objektum törlése szerkesztésnél



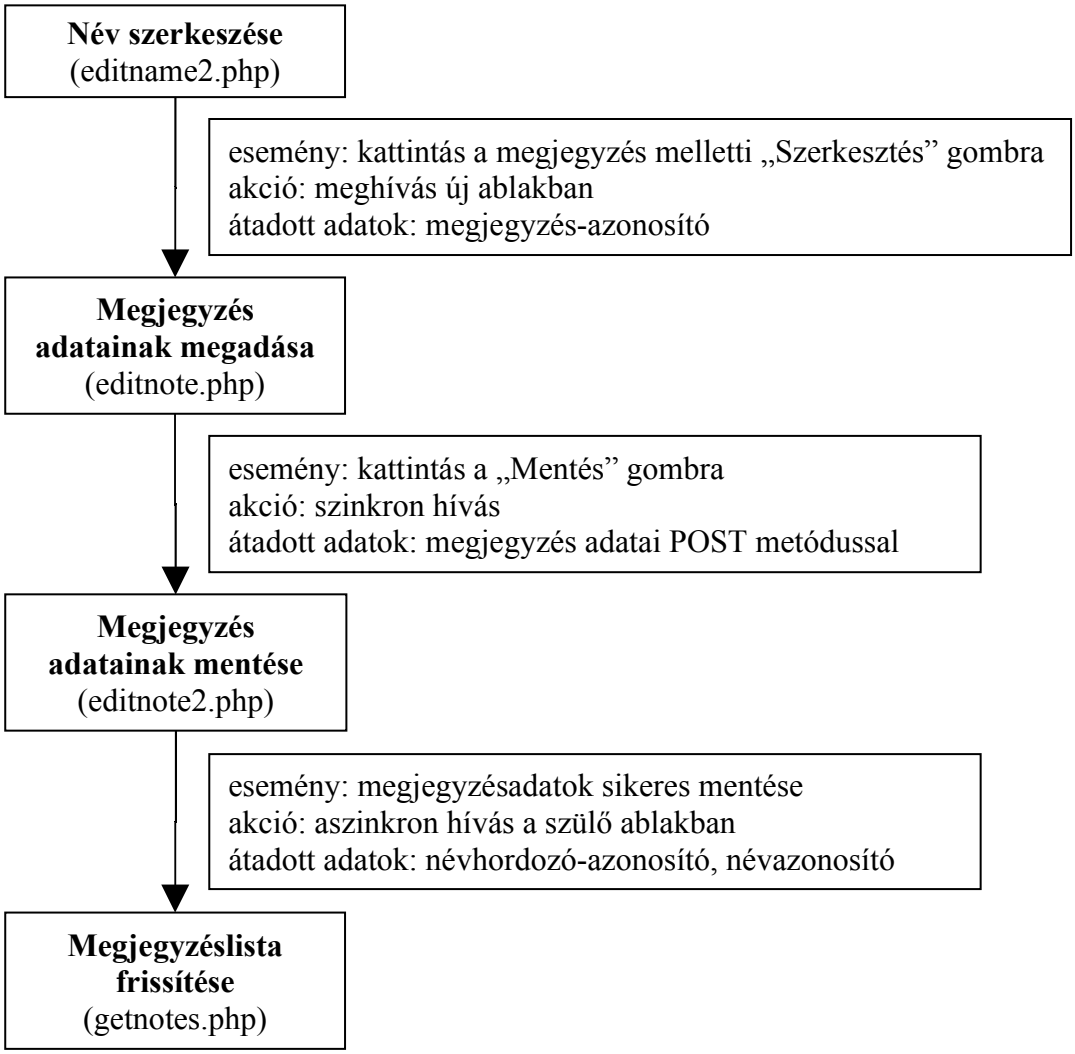
Ha a névszerkesztésnél kapcsolódó objektumot szeretnénk törölni, akkor a kapcsolódó objektum melletti „Törlés” feliratú gombra kattintva új ablakban megjelenik a kapcsolódó objektum törlésének megerősítésére szolgáló kérdés, melynek forráskódját a deleventrelobj.php fájl tartalmazza. A „Törlés” gomb megnyomására meghívódik a deleventrelobj2.php fájl mely a kapcsolódó objektum törlését végzi. A törlendő kapcsolódó objektum azonosítója POST metódussal érkezik, ami alapján a program összeállítja az XML formátumú NDA-protokoll kérést. Ha a törlés sikeres volt, akkor a szerkesztő ablakban frissül a kapcsolódó objektumok listája, amelyet a getrelobjects.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.9. Megjegyzés felvitele szerkesztésnél



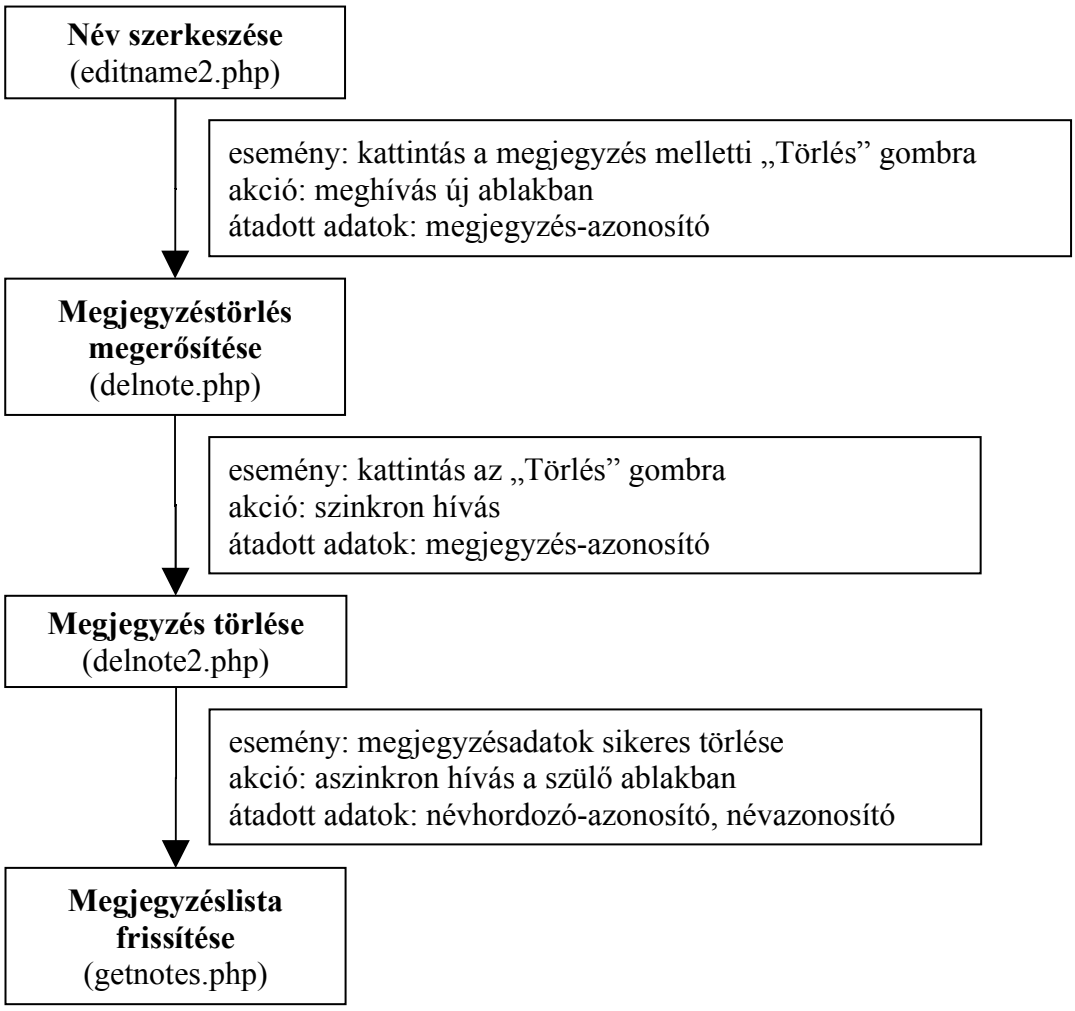
Ha a névszerkesztésnél új megjegyzést szeretnénk felvenni, akkor az „Új megjegyzés” feliratú gombra kattintva új ablakban megjelenik a megjegyzés adatainak megadására szolgáló űralp, melynek forráskódját az editnote.php fájl tartalmazza. A „Mentés” gomb megnyomására meghívódik az editnote2.php fájl mely a megjegyzésadatok mentését végzi. A megjegyzésadatok POST metódussal érkeznek, ami alapján a program összeállítja az XML formátumú NDA-protokoll kérést. Ha a mentés sikeres volt, akkor a szerkesztő ablakban frissül a megjegyzések listája, amelyet a getnotes.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.10. Megjegyzés módosítása szerkesztésnél



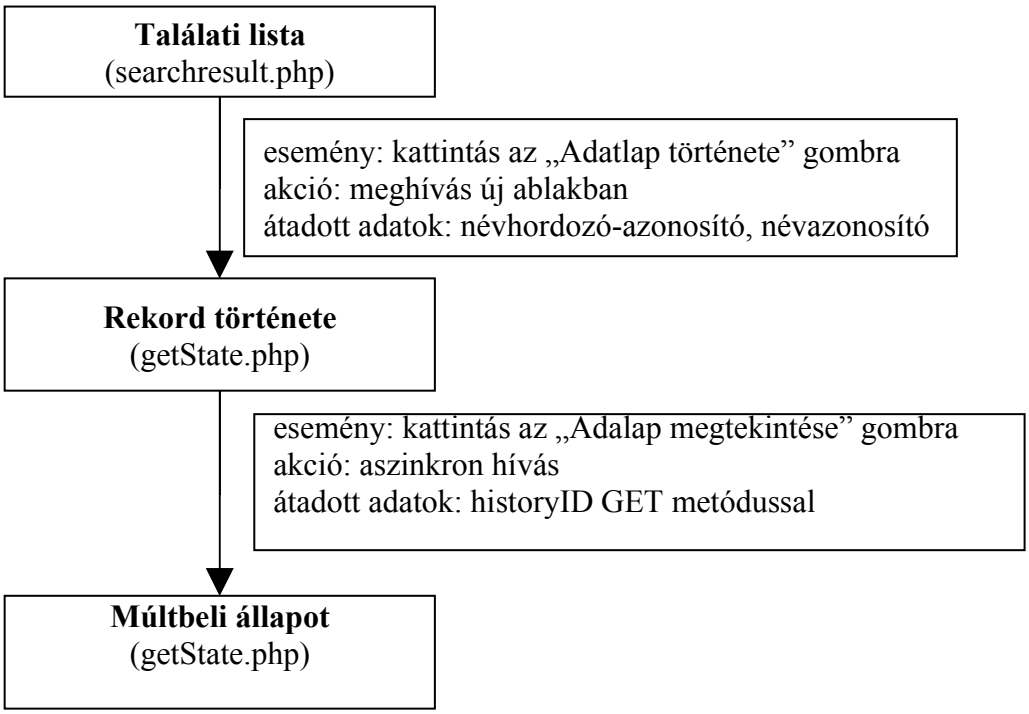
Ha a névszerkesztésnél új megjegyzést szeretnénk módosítani, akkor a megjegyzés melletti „Szerkesztés” feliratú gombra kattintva új ablakban megjelenik a megjegyzés adatainak meódosítására szolgáló űralp, melynek forráskódját az editnote.php fájl tartalmazza. A „Mentés” gomb megnyomására meghívódik az editnote2.php fájl mely a megjegyzésadatok mentését végzi. A megjegyzésadatok POST metódussal érkeznek, ami alapján a program öszzeállítja az XML formátumú NDA-protokoll kérést. Ha a mentés sikeres volt, akkor a szerkesztő ablakban frissül a megjegyzések listája, amelyet a getnotes.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.4.11. Megjegyzés törlése szerkesztésnél



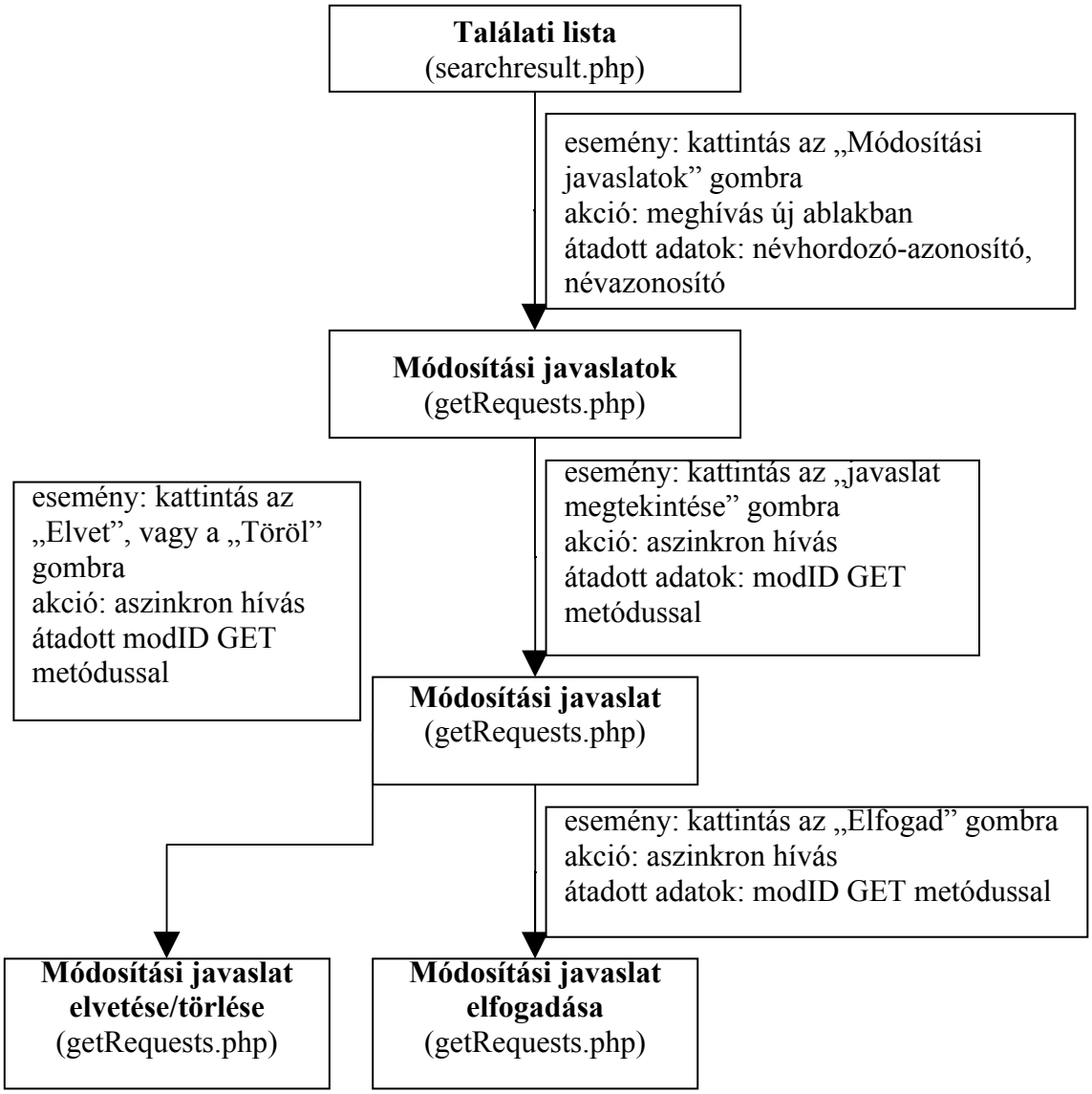
Ha a névszerkesztésnél megjegyzést szeretnénk törölni, akkor a megjegyzés melletti „Törlés” feliratú gombra kattintva új ablakban megjelenik a törlés megerősítésére vonatkozó kérdés, melynek forráskódját a delnote.php fájl tartalmazza. A „Törlés” gomb megnyomására meghívódik a delnote2.php fájl mely a megjegyzés törlését végzi. A megjegyzés-azonosító POST metódussal érkeznek, ami alapján a program összeállítja az XML formátumú NDA-protokoll kérést. Ha a törlés sikeres volt, akkor a szerkesztő ablakban frissül a megjegyzések listája, amelyet a getnotes.php program állít elő szintén egy XML formátumú NDA-protokoll kérés alapján.

2.5. Adatlap története



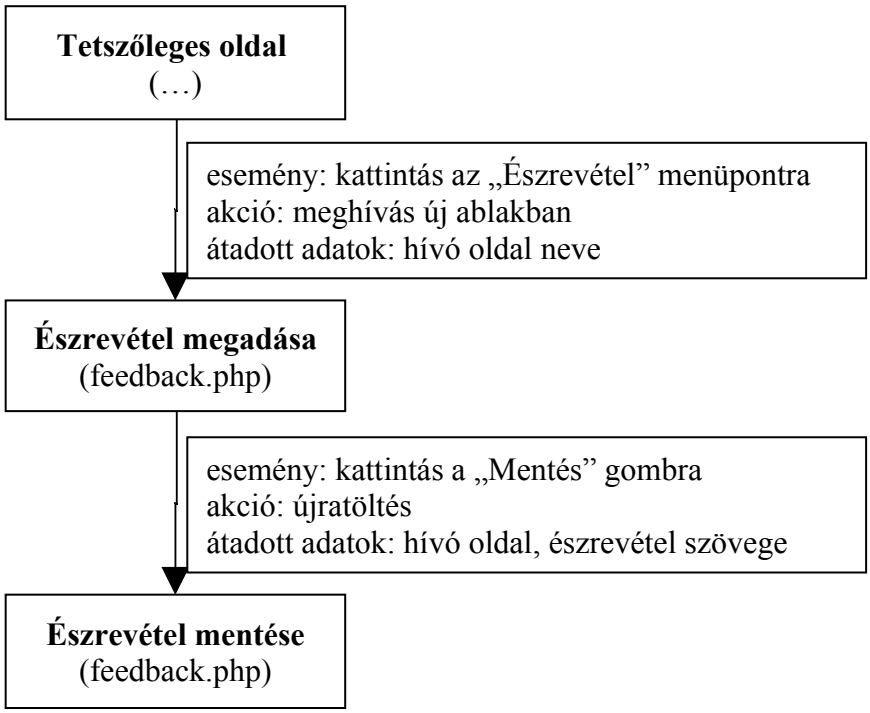
A nevek adatainak történetét csak regisztrált névtér adatgazda tudja megtekinteni, az után, hogy sikeresen bejelentkezett. Ebben az esetben a keresési találati listában megjelenő „Adatlap története” feliratra kattintva érhető el az adatlap változási időpontjait felsoroló történeti lista. A lista egy XML formátumú NDA-protokoll kérés eredményeképpen áll elő. Ha a GET változóban a historyID változó értéke nem üres, akkor a program szintén az NDA-protokollon keresztül lekéri a múltbeli állapotot, amely aszinkron hívással töltődik be a megfelelő HTML div elembe.

2.6. Módosítási javaslatok



Egy név adataira vonatkozó módosítási javaslatokat csak az a regisztrált névtér adatgazda tudja megtekinteni, aki az adott nevet felvette a rendszerbe, és ő is csak az után, hogy sikeresen bejelentkezett. A módosítási javaslatokat a getRequests.php fájl jeleníti meg, úgy hogy a GET paraméterben érkező névhordozó- és névazonosító alapján előállítja a javaslatokat lekérő XML formátumú NDA-protokoll kérést. A szintén XML formátumú válasz alapján ekkor kilstázza a módosítási javaslatok dátumait. A „javaslat megtekintése” gombra kattintva jelenik meg maga a módosítási javaslat, amelynek adatait szintén getRequests.php fájl kéri le az NDA-protokollon keresztül egy aszinkron http hívással, aminek eredményét a megfelelő html div elembe írja. A módosítási javaslat elfogadható, elvethető, vagy törölhető. Mindhárom műveletet a getRequests.php fájlban lévő php kód valósítja meg, úgy hogy a GET paraméterben kapott modID alapján legenerálja a megfelelő NDA-protokoll kérést.

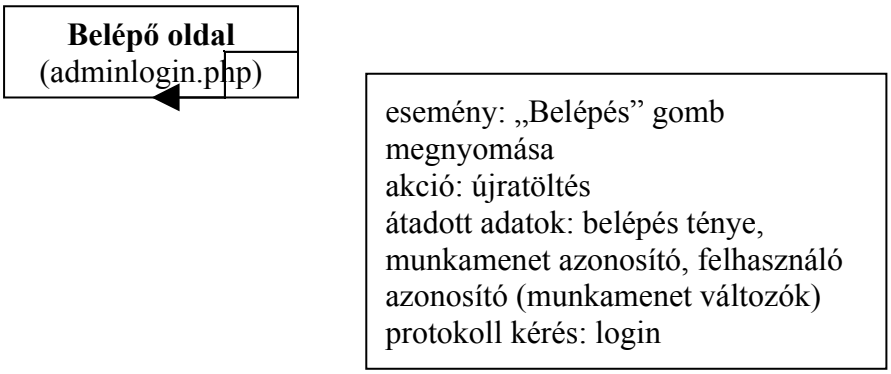
2.7. Észrevétel küldése



Az észrevételküldést működtető program forráskódját a feedback.php fájl tartalmazza. Mentéskor a hívó oldal neve GET paraméterben az észrevétel szövege POST metódussal adódik át.

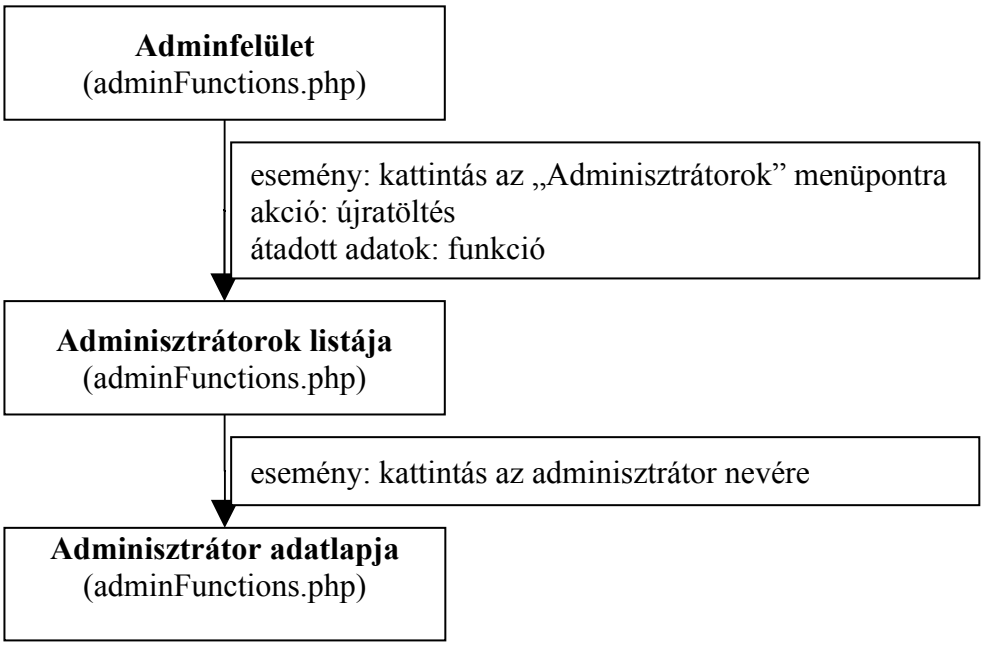
2.8. Adminisztrátori funkciók

2.8.1. Bejelentkezés



Az adminisztrátorok az adminlogin.php oldalon tudnak belépni az adminfelületre, felhasználónév és jelszó megadásával. A „Belépés” gomb megnyomására az oldal újratöltődik. Az újratöltés után a $_POST változóban érkező felhasználónév és jelszó alapján a program összeállítja az XML szerkezetű login NDA-protokoll kérést, a kérés szintén XML szerkezetű válasza alapján pedig a PHP munkamenet változóban eltárolja a bejelentkezés tényét, a munkamenet azonosítót és a felhasználó azonosítót. Sikeres bejelentkezés esetén a program betölti a keresőoldal, sikertelen bejelentkezésnél újra a bejelentkező űrlap jelenik meg.

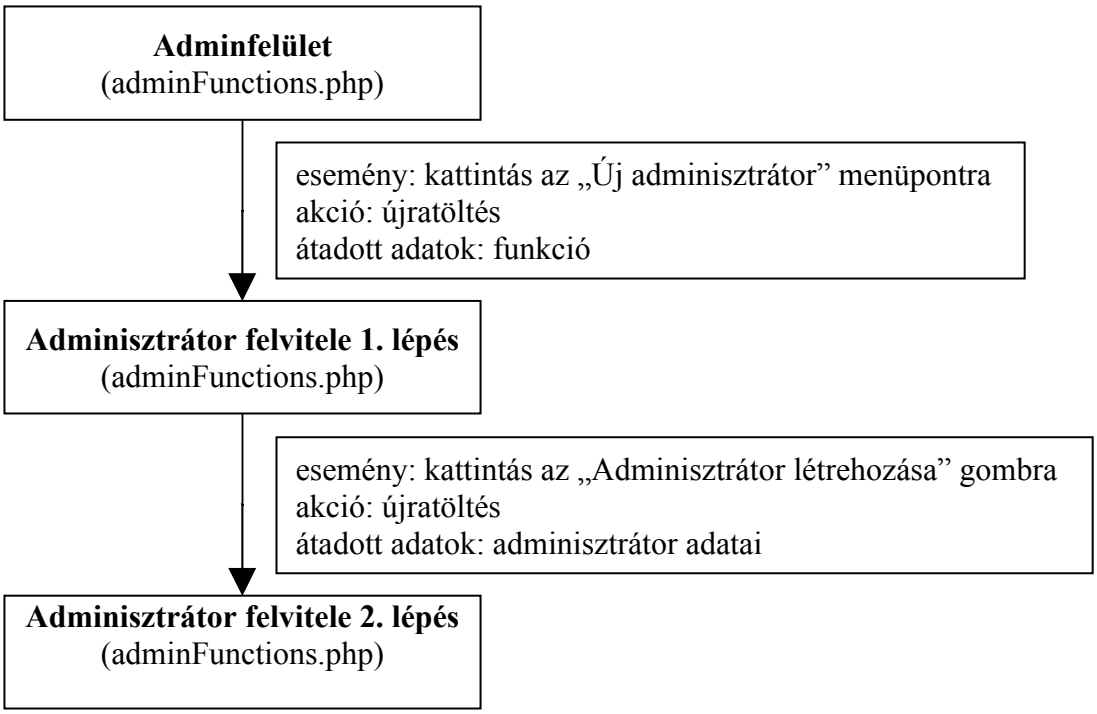
2.8.2. Adminisztrátorok listája



Az adminisztrátorok listáját az adminFunctions.php fájl jeleníti meg, ha az adminisztrátori felületen az „Adminisztrátorok” menüpontra kattintunk. A funkció neve POST metódussal adódik át az oldal újratöltésekor. Az egyes adminisztrátorok adatai közvetlen

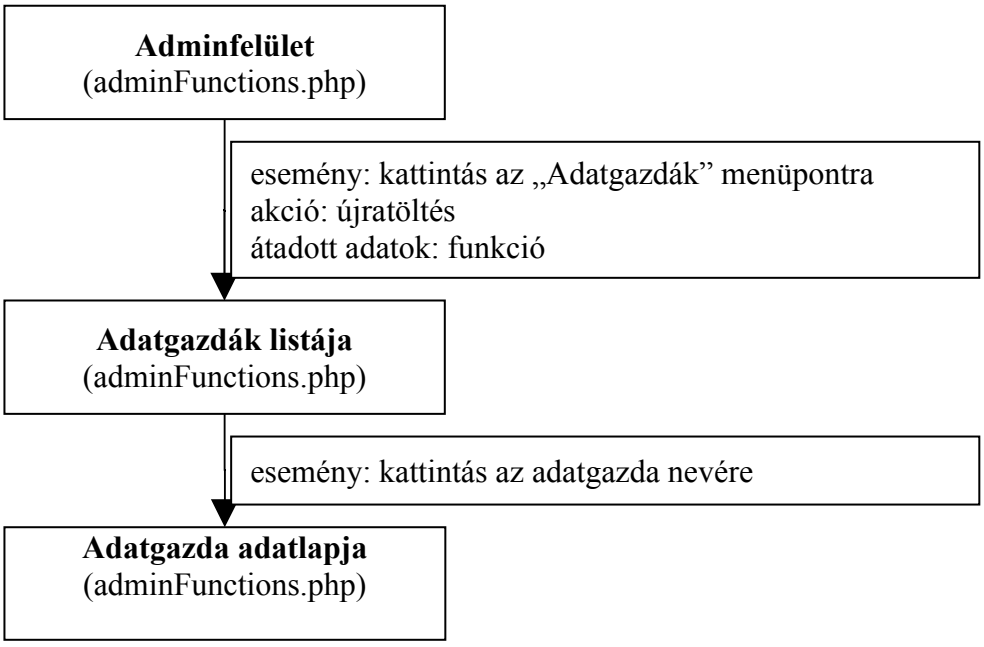
adatbázislekérdezés útján átadódnak az oldalnak és rejtett html div elemekre íródnak, melyek az adott listaelemre kattintva jelennek meg.

2.8.3. Adminisztrátor felvétele



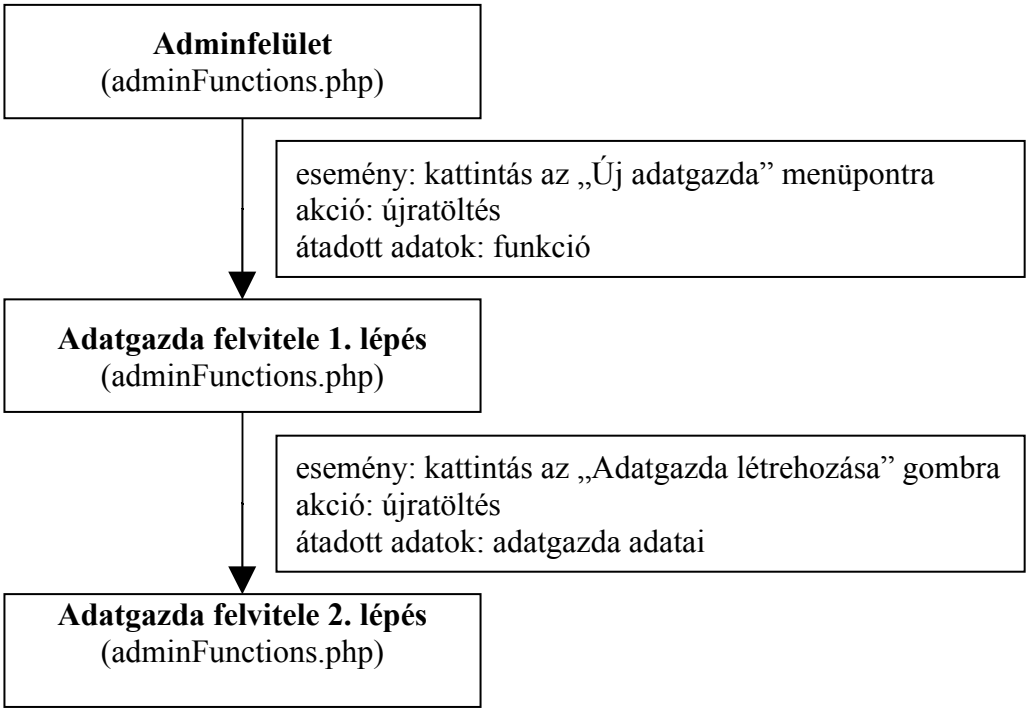
Az adminisztrátorok felvitelét végző program forráskódját az adminFunctions.php fájl tartalmazza. Az „Új adminisztrátor” menüpontra kattintva megjelenik az adatok felvitelét lehetővé tévő űrlap. Az „Adminisztrátor létrehozása” gombra kattintva az oldal újratöltödik, és az új adminisztrátor adatai POST metódussal adódnak tovább, hogy a program elmentse őket az adatbázisba.

2.8.4. Adatgazdák listája



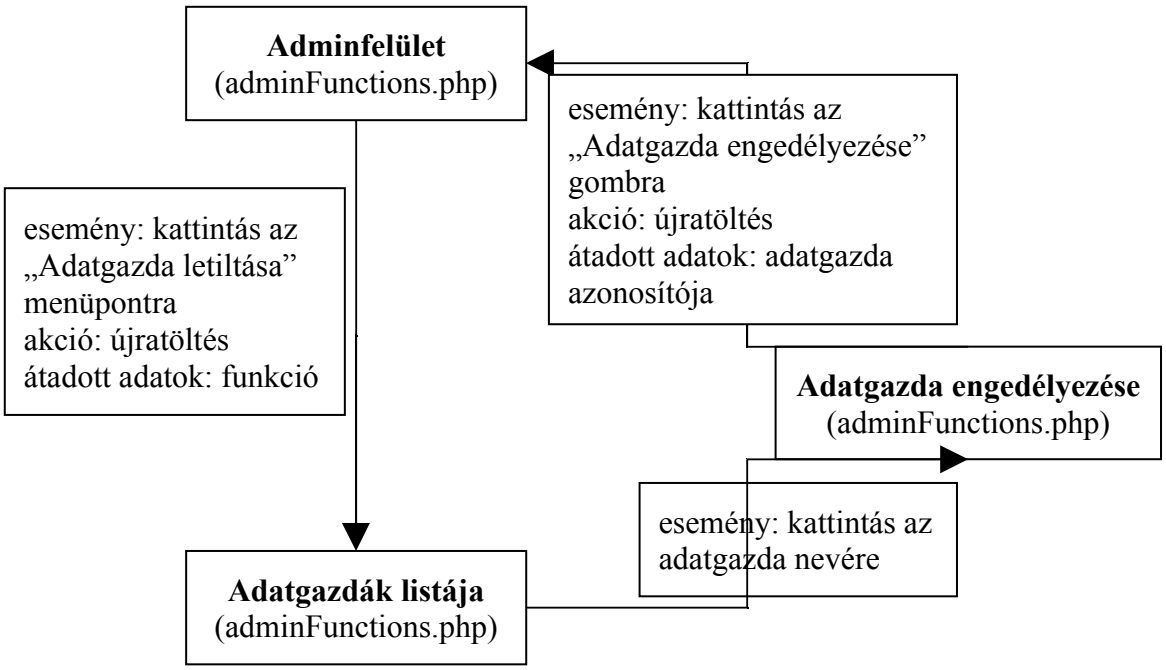
Az adatgazdák listáját az adminFunctions.php fájl jeleníti meg, ha az adminisztrátori felületen az „Adatgazdák” menüpontra kattintunk. A funkció neve POST metódussal adódik át az oldal újratöltésekor. Az egyes adatgazdák adatai közvetlen adatbázislekérdezés útján átadódnak az oldalnak és rejtett html div elemekre íródnak, melyek az adott listaelemre kattintva jelennek meg.

2.8.5. Adatgazda felvétele



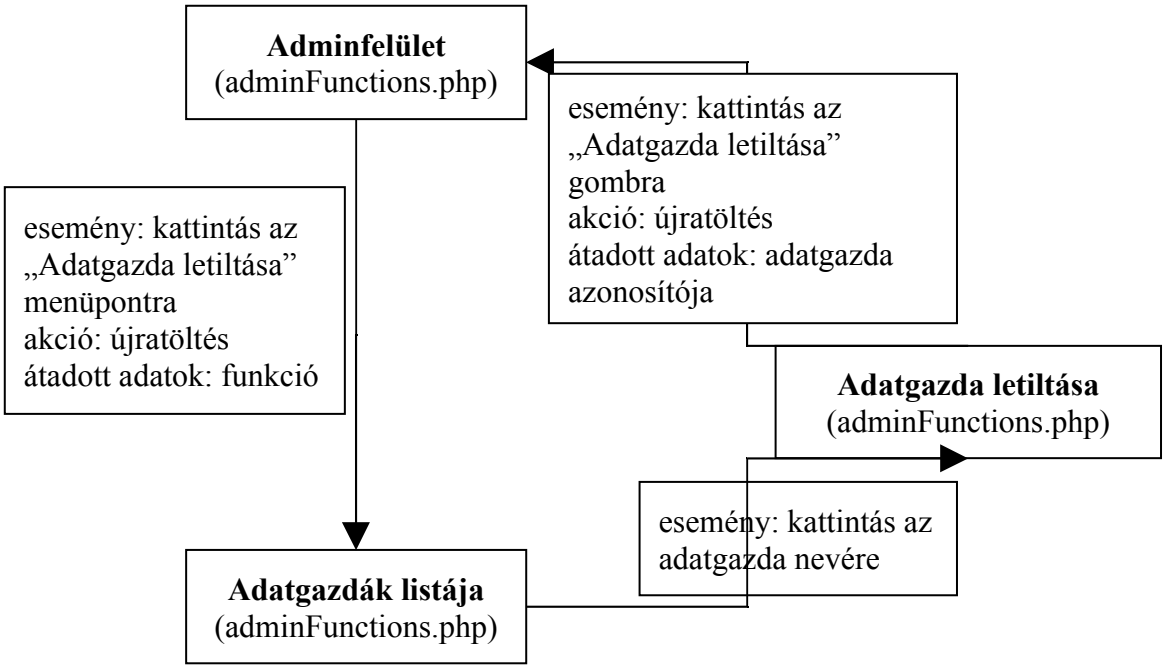
Az adatgazdák felvitelét végző program forráskódját az adminFunctions.php fájl tartalmazza. Az „Új adatgazda” menüpontra kattintva megjelenik az adatok felvitelét lehetővé tévő űrlap. Az „Adatgazda létrehozása” gombra kattintva az oldal újratöltődik, és az új adatgazda adatai POST metódussal adódnak tovább, hogy a program elmentse őket az adatbázisba.

2.8.6. Adatgazda engedélyezése



Az adatgazdák engedélyezését végző program forráskódját az adminFunctions.php fájl tartalmazza. Az „Adatgazda letiltása” menüpontra kattintva megjelenik az adatgazdák listája. Az engedélyezni kívánt adatgazda nevére kattintva megnyílik annak adatlapja és ha nincs engedélyezve, akkor az „Adatgazda engedélyezése” gomb. Erre a gombra kattintva az oldal újratöltődik és az adatgazda státusza engedélyezettre változik.

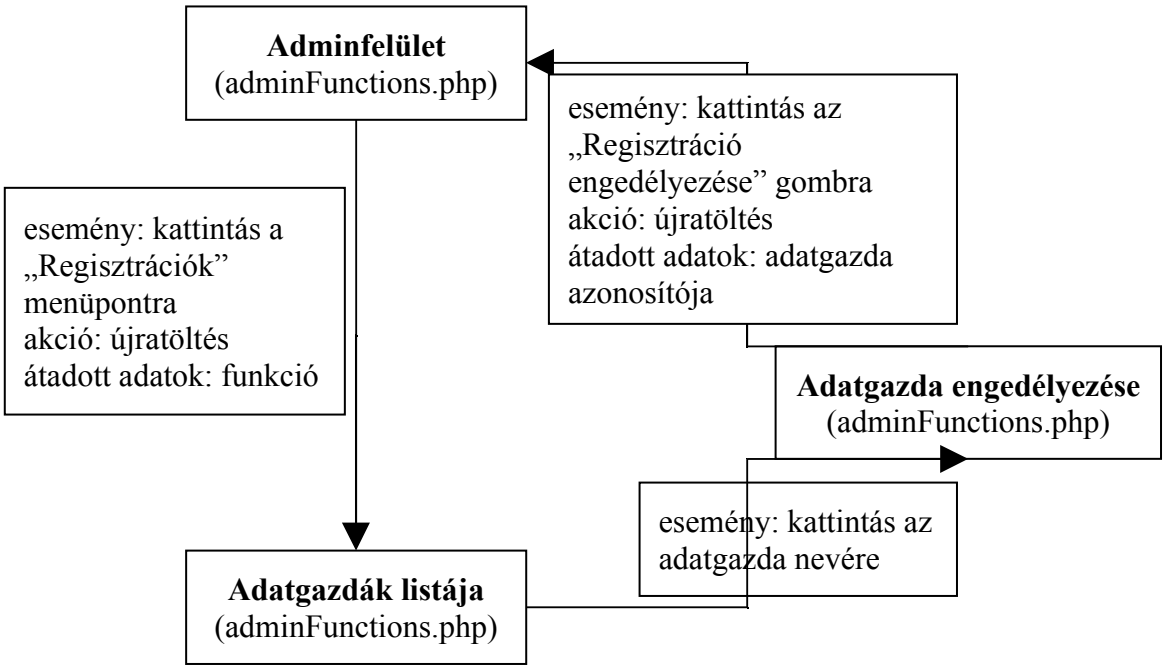
2.8.7. Adatgazda letiltása



Az adatgazdák letiltását végző program forráskódját az adminFunctions.php fájl tartalmazza. Az „Adatgazda letiltása” menüpontra kattintva megjelenik az adatgazdák listája. A letiltani kívánt adatgazda nevére kattintva megnyílik annak adatlapja, és ha nincs letiltva, akkor az

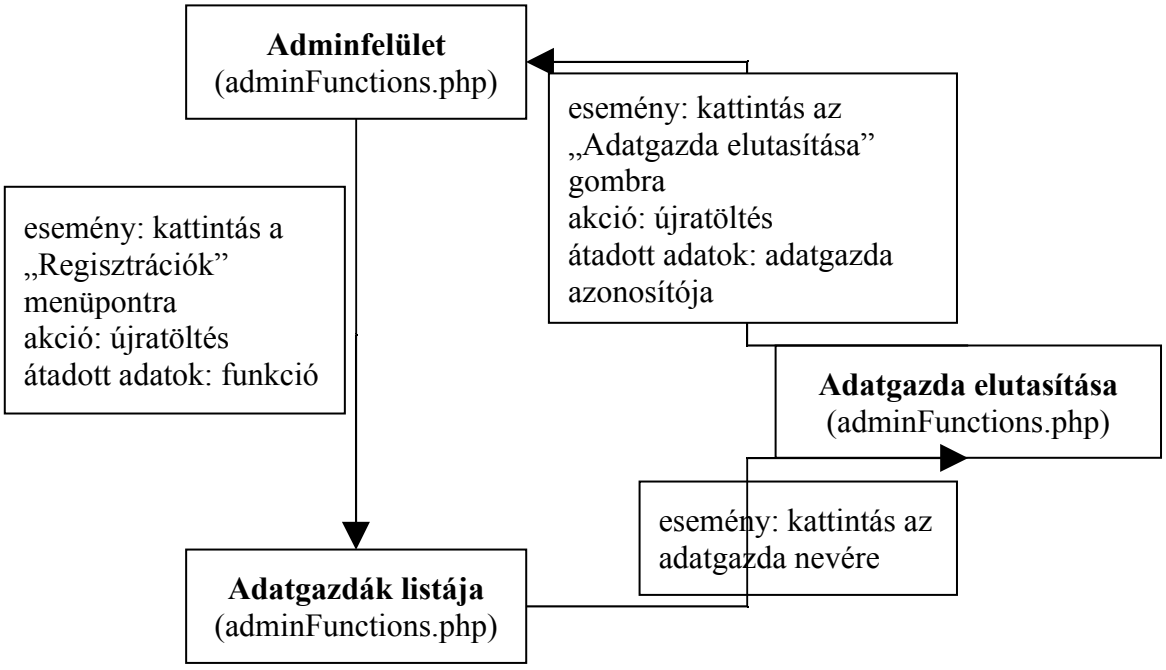
„Adatgazda letiltása” gomb. Erre a gombra kattintva az oldal újratöltődik és az adatgazda státusza tiltottra változik.

2.8.8. Regisztráció engedélyezése



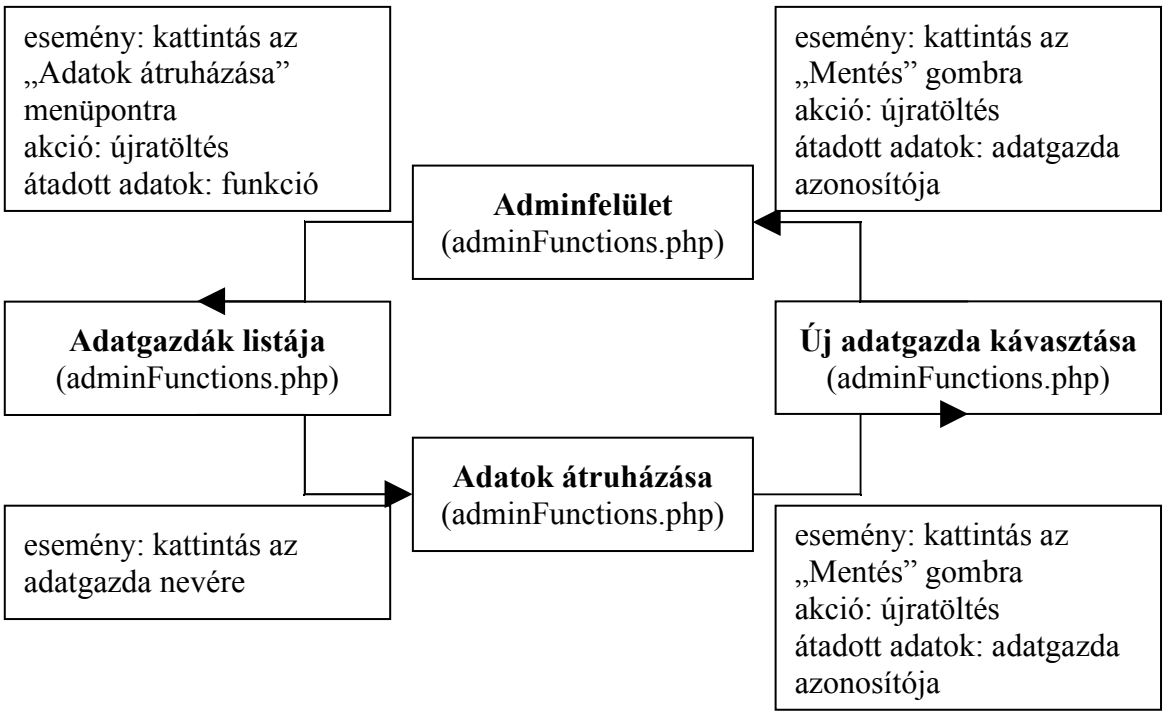
Az adatgazdák engedélyezését végző program forráskódját az adminFunctions.php fájl tartalmazza. Az „Regisztrációk” menüpontra kattintva megjelenik a frissen regisztrált adatgazdák listája. Az engedélyezni kívánt adatgazda nevére kattintva megnyílik annak adatlapja, és a „Regisztráció engedélyezése” gomb. Erre a gombra kattintva az oldal újratöltődik és az adatgazda státusza engedélyezettre változik.

2.8.9. Regisztráció elutasítása



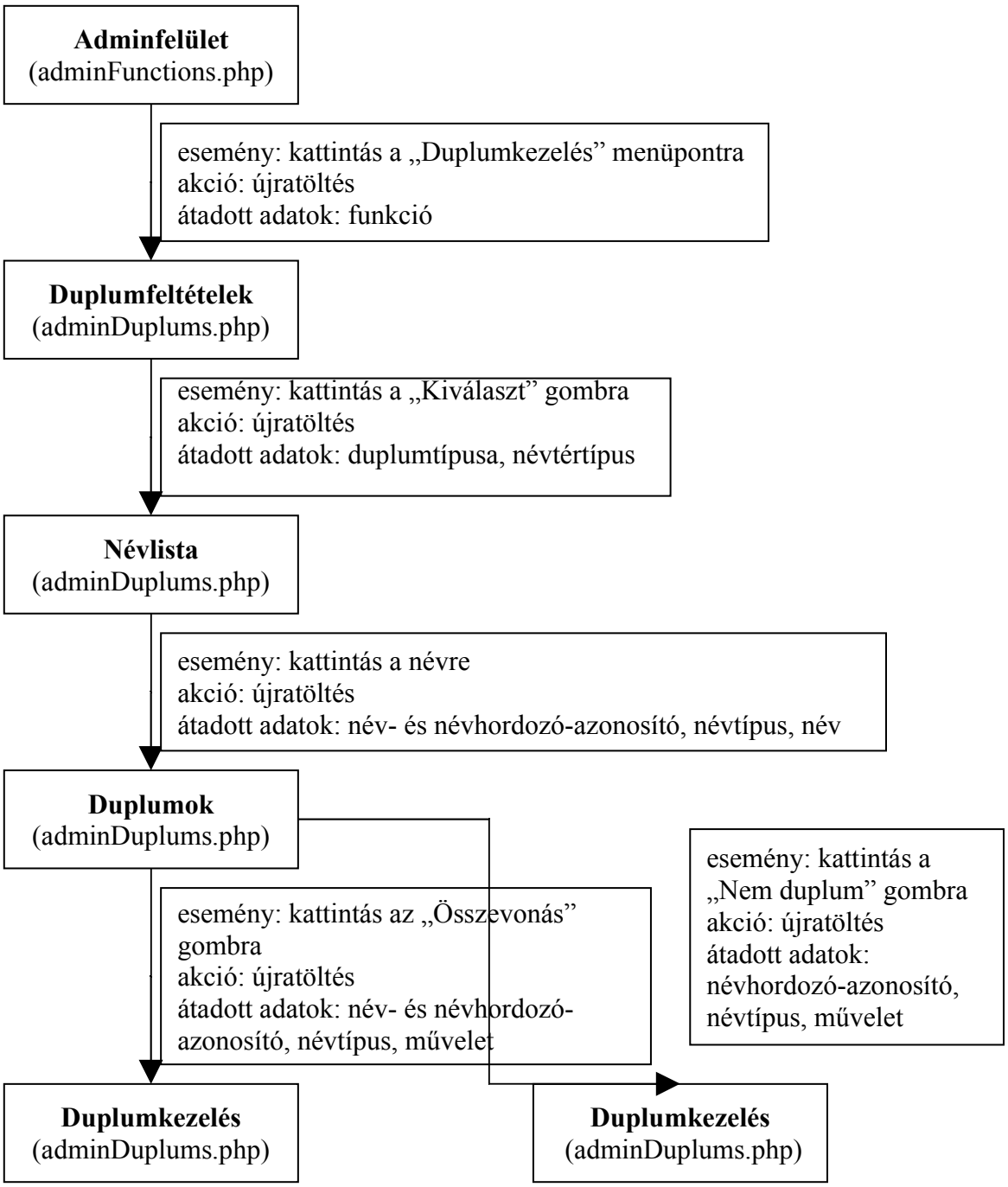
Az adatgazdák letiltását végző program forráskódját az adminFunctions.php fájl tartalmazza. Az „Adatgazda letiltása” menüpontra kattintva megjelenik az adatgazdák listája. A letiltani kívánt adatgazda nevére kattintva megnyílik annak adatlapja, és ha nincs letiltva, akkor az „Adatgazda letiltása” gomb. Erre a gombra kattintva az oldal újratöltődik és az adatgazda státusza tiltottra változik.

2.8.10. Adatgazda adatainak átruházása



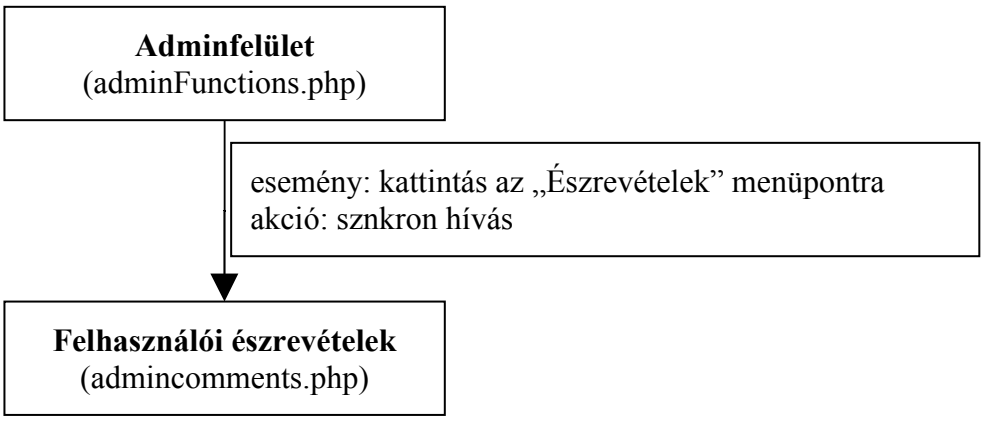
Az adatgazda adatainak átruházását végző program forráskódját az adminFunctions.php fájl tartalmazza. Az „Adatok átruházása” menüpontra kattintva megjelenik a letiltott adatgazdák listája. A kívánt adatgazda nevére kattintva megnyílik annak adatlapja, és az „Adatok átruházása” gomb. Erre a gombra kattintva az oldal újratöltődik és mejegelenik az engedélyezett státuszú adatgazdák listája, amiből kiválasztható, hogy ki legyen az új adattulajdonos. A „Mentés” gombra kattinva az adatok tulajdonosa megváltozik.

2.8.11. Duplumkezelés



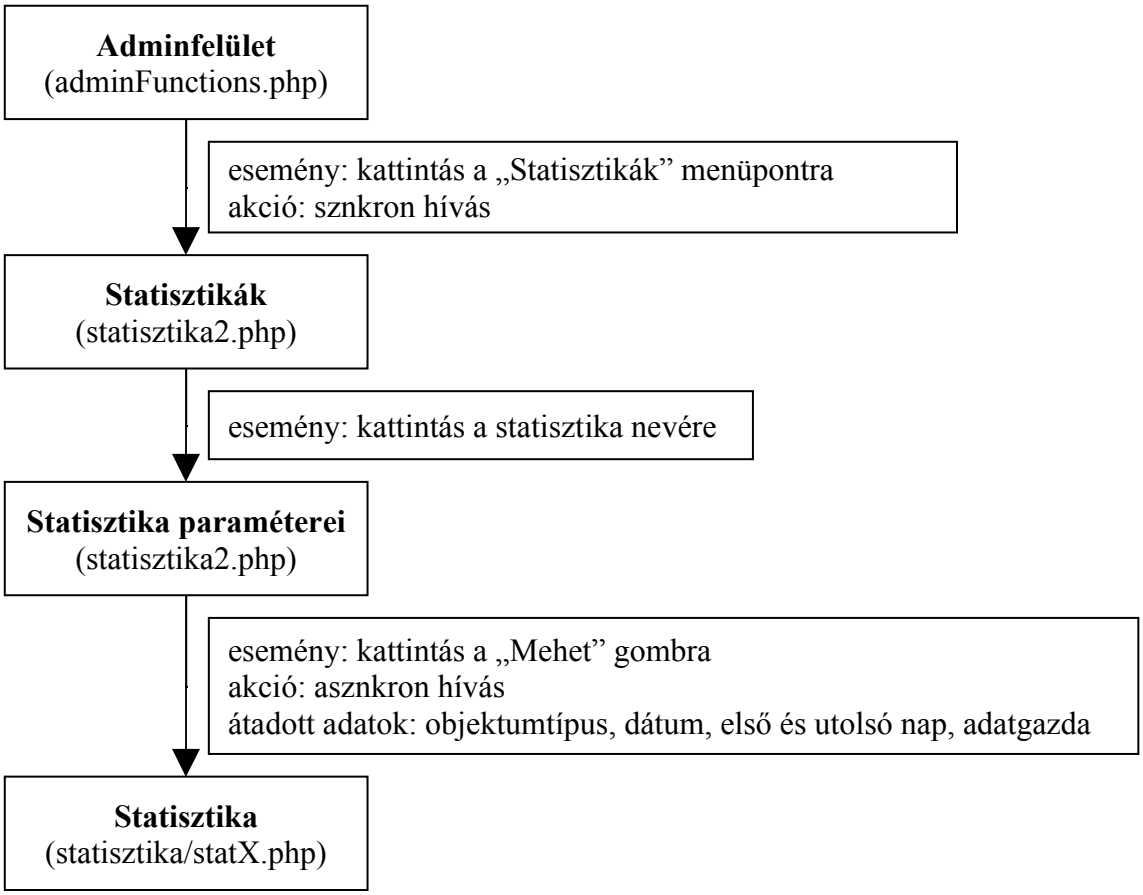
A duplumkezelés adminisztrátori funkciót működtető programkódot az adminDuplums.php fájl tartalmazza. Ez a funkció a „Duplumkezelés” menüpontra kattintva hívható meg. Első lépésben megjelenik a duplumellenőrzés paramétereinek megadására szolgáló űrlap, melyen megadható a duplumok típusa és a névtértípus. A „Kiválaszt” gomb megnyomására a beállított paramétereknek megfelelően, az oldal ujratöltődése után megjelenik a duplumok listája. A paraméterek POST metódussal adódnak át. A listában egy névre kattintva az oldal újratöltésével megjelenik a kiválasztott névhez tartozó duplumok listája. Itt a név- és névhordozó-azonosító, a névtípus és a név GET paraméterben adódnak át. Az „Összevonás” gombra kattintva újratöltődik az oldal és a név- és névhordozó-azonosító, a névtípus és a művelet POST metódussal átadódnak a programnak. Újratöltés után a program meghívja az összevonást kezdeményező XML formátumú NDA-protokoll kérést.

2.8.12. Észrevételek



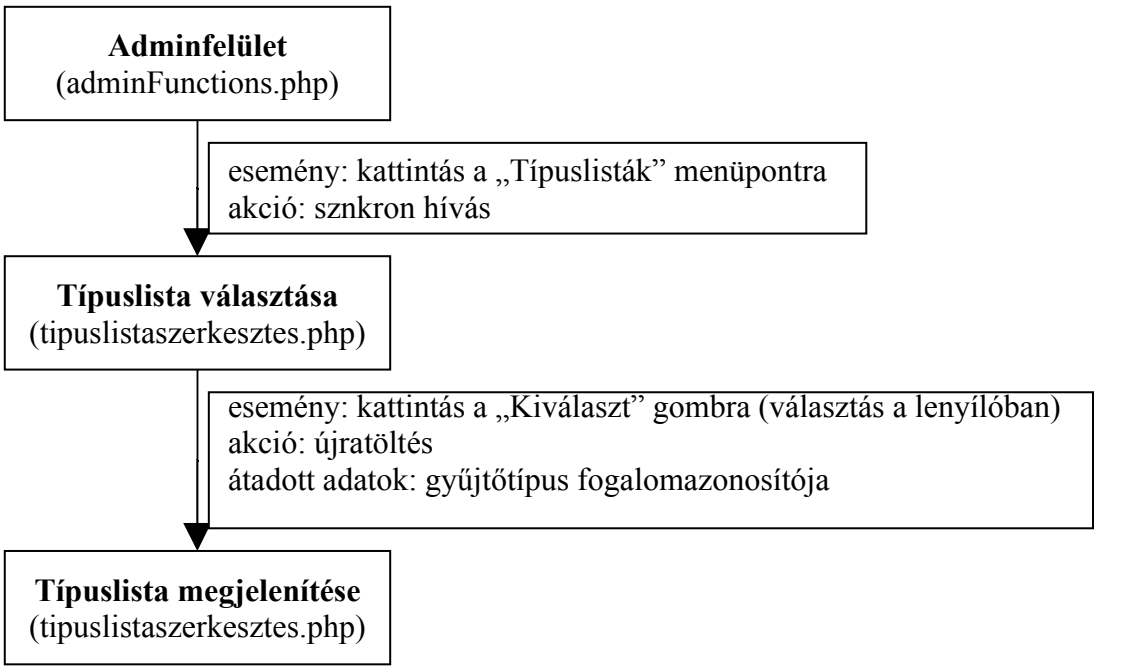
A felhasználói észrevételeket az admincomments.php fájl jeleníti meg, ha az adminisztrátori felületen az „Észrevételek” menüpontra kattintunk. A program közvetlenül az adatbázisból kérdezi le az észrevétellistát.

2.8.13. Statisztikák



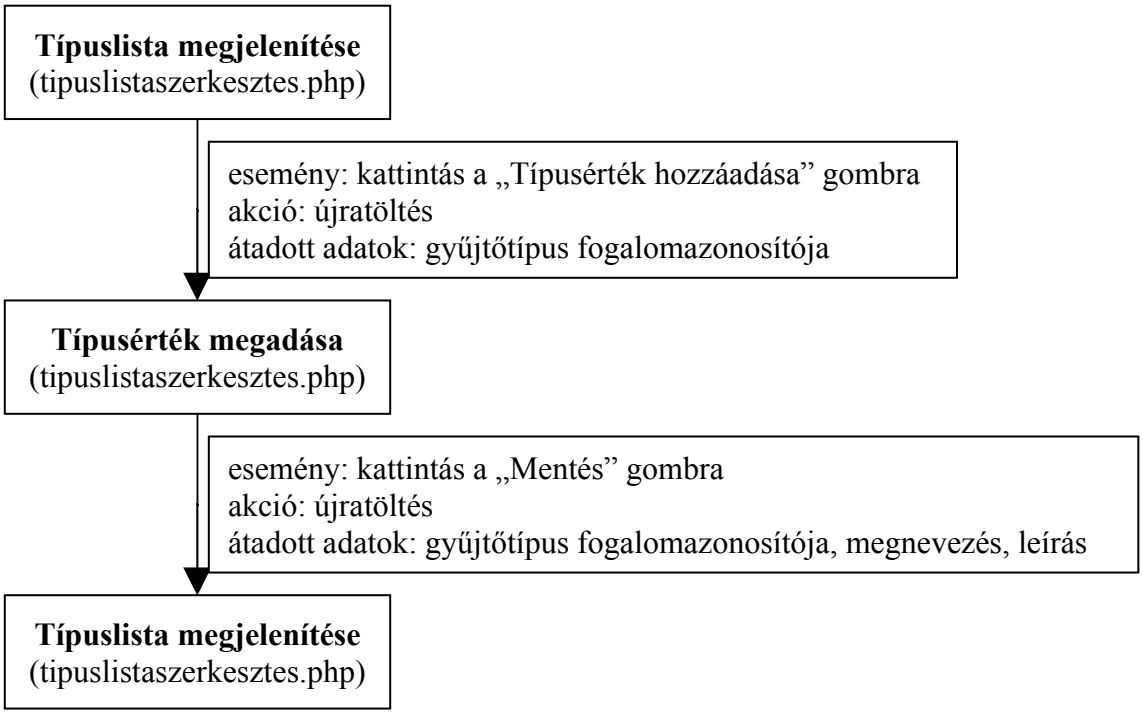
A statisztikák megjelenítését végző program forráskódját a statisztika2.php fájl tartalmazza. A statisztika paramétereinek megadása utána a „Mehet” gombra kattintva lefut az adott statisztikához tartozó javascript függvény. A függvény aszinkron http hívással letölti a statX.php fájlt, amely megjeleníti a statisztikai eredményeket. A statX.php fájlnévben az X a statisztika sorszámát jelöli általánosan. Ez a fájl GET változóban kapja meg a statisztika paramétereit.

2.8.14. Típuslisták szrkesztése



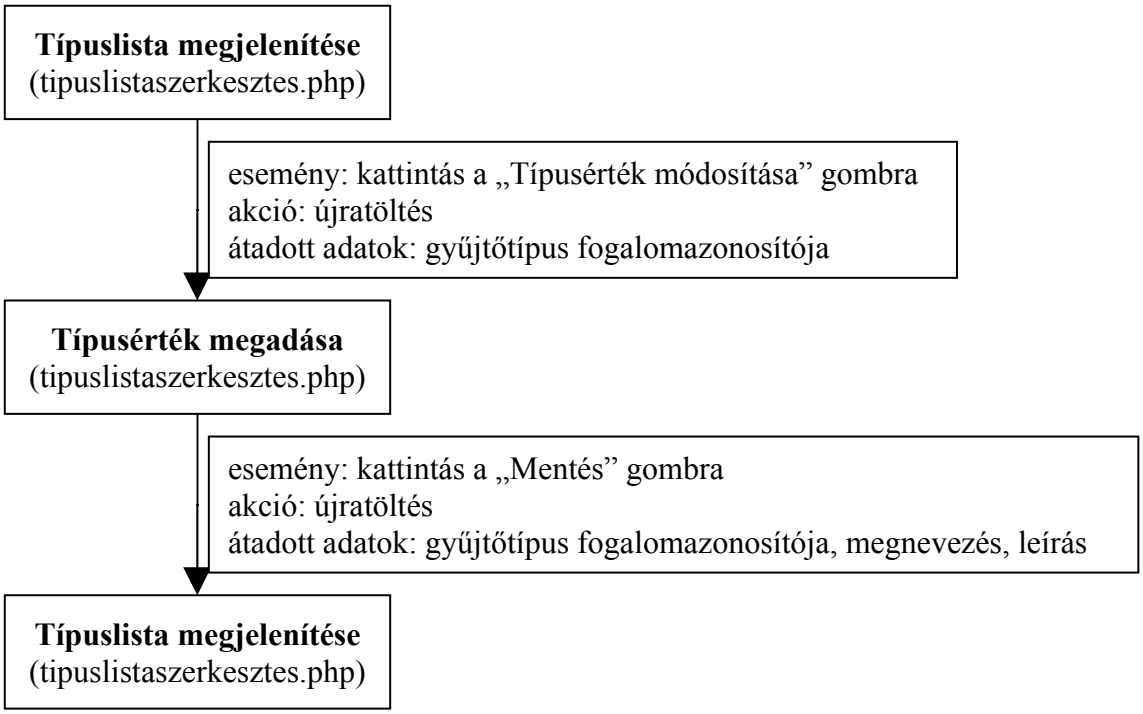
A típuslisták szrkesztését végző program forráskódját a tipuslistaszerkesztes.php fájl tartalmazza. A típuslista-szerkesztés első lépése a gyűjtőtípus fogalom kiválasztása egy lenyíló listából. Ennek eredményeképpen az oldal újratöltődik és a fogalomazonosító GET paraméterben átadódik. Az oldal újratöltése után megjelenik a kiválasztott gyűjtőtípus alá tartozó típusértékek listája és a hozzáadás, módosítás, törlés gombok.

2.8.14.1 Típusérték hozzáadása



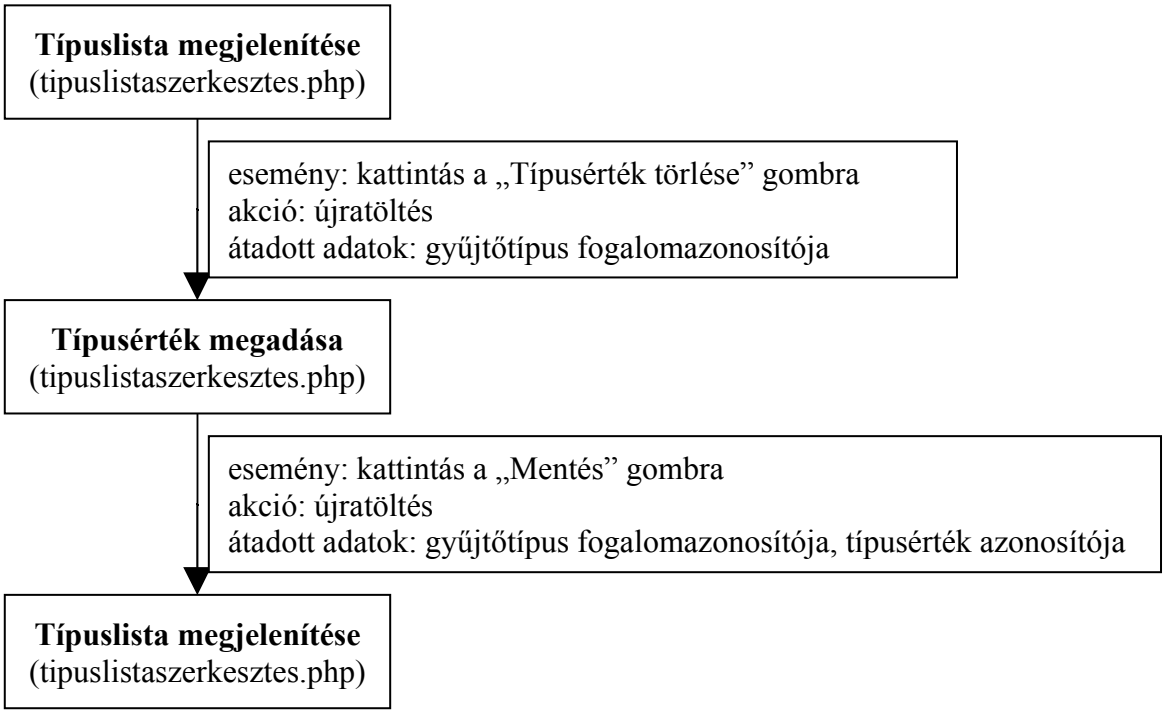
A típusérték hozzáadásánál, az új elem megnevezése és leírása az oldal újratöltődésekor POST metódussal adódik tovább, hogy a program mentse őket az adatbázisba.

2.8.14.2 **Típusérték módosítása**



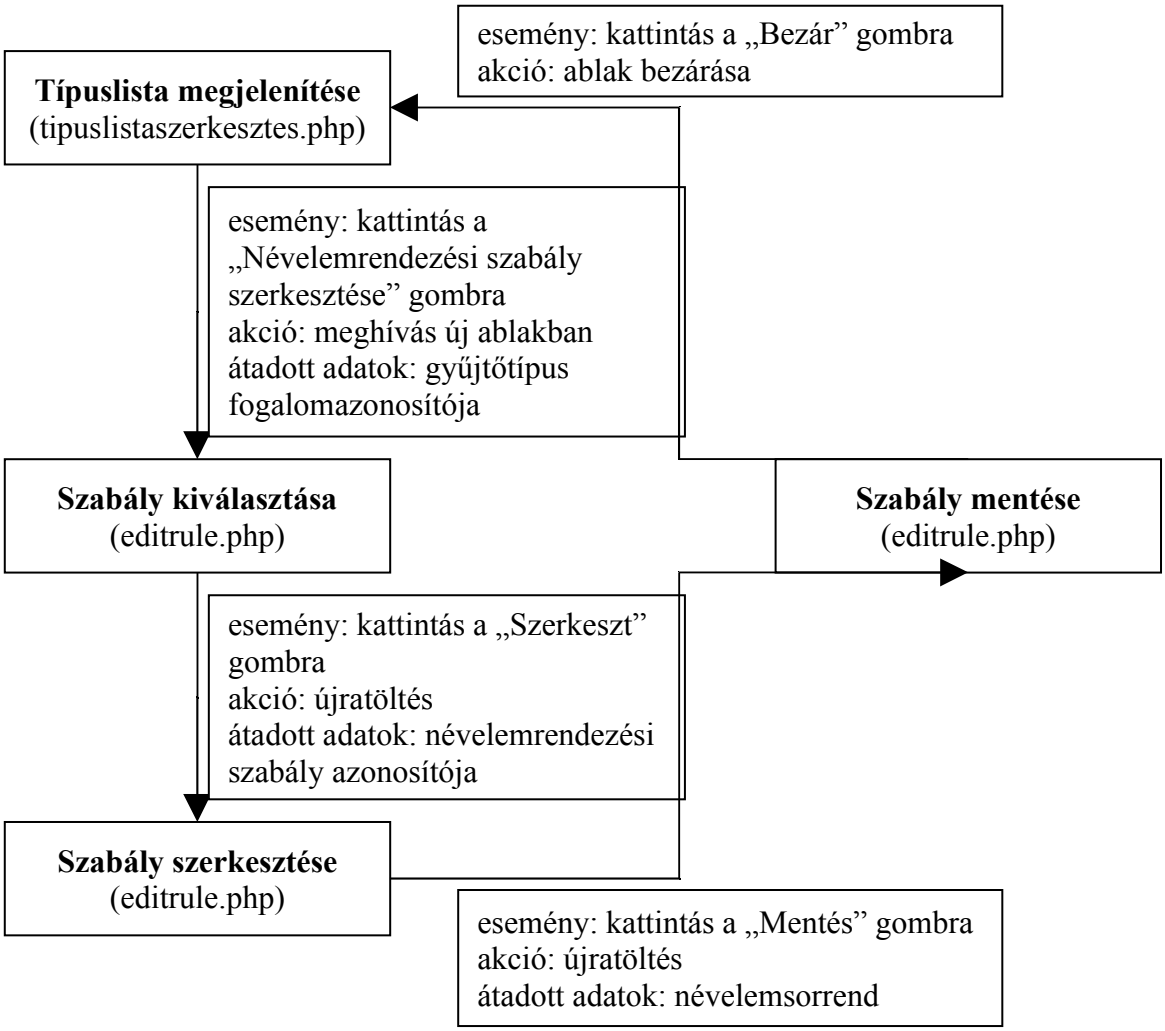
A típusérték módosításánál, a szerkesztett elem megnevezése és leírása az oldal újratöltődésekor POST metódussal adódik tovább, hogy a program mentse őket az adatbázisba.

2.8.14.3 **Típusérték törlése**



A típusérték törlésénél, az eltávolítandó elem azonosítója az oldal újratöltődésekor POST metódussal adódik tovább, hogy a program törölje az adatbázisból.

2.8.14.4 **Névelemrendezési szabály szerkesztése**



A névelemrendezési szabály szerkesztését végző program forráskódját az editrule.php fájl tartalmazza. A névelemrendezési szabály szerkesztésének első lépése a szabály kiválasztása egy lenyíló listából. Ennek eredményeképpen az oldal újratöltődik és a fogalomazonosító POST paraméterben átadódik. Az oldal újratöltése után megjelenik a névelemek listája, amelyben módosítható a sorrend. A mentés gombra kattintva az oldal újratöltődik és a névelemek sorrendje elmentődik az adatbázisba.

3. Adatbázis (create szkriptek)

3.1. ADMINS2

Az adminisztrátorok adatait tartalmazó tábla

```
CREATE TABLE admins2 (id NUMBER NOT NULL, name VARCHAR2(250), address
VARCHAR2(250), tel VARCHAR2(50), email VARCHAR2(50), contactperson
VARCHAR2(100), username VARCHAR2(20), password VARCHAR2(50), regdate DATE);
CREATE SEQUENCE users2_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE admins2 ADD CONSTRAINT pk_admins2 PRIMARY KEY (id);
CREATE INDEX admins2_name ON admins2(name);
CREATE INDEX admins2_username ON admins2(username);
CREATE INDEX admins2_password ON admins2(password);
CREATE INDEX admins2_regdate ON admins2(regdate);
commit;
```

3.2. USERS2

Az adatgazdák adatait tartalmazó tábla.

```
CREATE TABLE users2 (id NUMBER NOT NULL, name VARCHAR2(250), address
VARCHAR2(250), tel VARCHAR2(50), email VARCHAR2(50), url VARCHAR2(200),
contactperson VARCHAR2(100), username VARCHAR2(20), password VARCHAR2(50),
status NUMBER, regdate DATE);
ALTER TABLE users2 ADD CONSTRAINT pk_users2 PRIMARY KEY (id);
CREATE INDEX users2_name ON users2(name);
CREATE INDEX users2_username ON users2(username);
CREATE INDEX users2_password ON users2(password);
CREATE INDEX users2_regdate ON users2(regdate);
commit;
```

3.3. USERFEEDBACK

Az adatgazdáknak a webes felülettel kapcsolatos észrevételeit tartalmazó tábla.

```
CREATE TABLE userfeedback (id NUMBER NOT NULL, feedbackdate DATE, ownerid
NUMBER, feedbacktext VARCHAR2(2000), page VARCHAR2(500));
CREATE SEQUENCE userfeedback_id_seq MINVALUE 1 START WITH 1 INCREMENT
BY 1;
ALTER TABLE userfeedback ADD CONSTRAINT pk_userfeedback PRIMARY KEY (id);
CREATE INDEX userfeedback_feedbackdate ON usercomments(feedbackdate);
commit;
```

3.4. LANGUAGE2

A nyelvek adatait tartalmazó tábla.

```
CREATE TABLE language2 (id NUMBER NOT NULL, name VARCHAR2(100),
technicalname VARCHAR2(100), description VARCHAR2(1024));
CREATE SEQUENCE language2_id_seq MINVALUE 1 START WITH 1 INCREMENT
BY 1;
ALTER TABLE language2 ADD CONSTRAINT pk_language2 PRIMARY KEY (id);
CREATE INDEX language2_name ON language2(name);
commit;
```

3.5. CHARCODESET

A karakterkódolások adatait tartalmazó tábla.

```
CREATE TABLE charcodeset (id NUMBER NOT NULL, name VARCHAR2(100),
technicalname VARCHAR2(100), description VARCHAR2(1024));
CREATE SEQUENCE charcodeset_id_seq MINVALUE 1 START WITH 1 INCREMENT
BY 1;
ALTER TABLE charcodeset ADD CONSTRAINT pk_charcodeset PRIMARY KEY (id);
CREATE INDEX charcodeset_name ON charcodeset(name);
commit;
```

3.6. WRITINGSYSTEM

Az írásrendszerek adatait tartalmazó tábla.

```
CREATE TABLE writingsystem (id NUMBER NOT NULL, name VARCHAR2(100),
technicalname VARCHAR2(100), description VARCHAR2(1024));
CREATE SEQUENCE writingsystem_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE writingsystem ADD CONSTRAINT pk_writingsystem PRIMARY KEY
(id);
CREATE INDEX writingsystem_name ON writingsystem(name);
commit;
```

3.7. CONCEPT

A fogalmak azonosítót és egymással való kapcsolatát leíró tábla.

```
CREATE TABLE concept (id NUMBER NOT NULL, coverconceptid NUMBER);
CREATE SEQUENCE concept_id_seq MINVALUE 1 START WITH 1 INCREMENT BY
1;
ALTER TABLE concept ADD CONSTRAINT pk_concept PRIMARY KEY (id);
commit;
```

3.8. LEXEME

A fogalmak megnevezéseit tartalmazó tábla.

```
CREATE TABLE lexeme (id NUMBER NOT NULL, conceptid NUMBER, name
VARCHAR2(1000), disambiguator VARCHAR2(250), pronunciation VARCHAR2(250),
languageid NUMBER, writingsystemid NUMBER, charcodesetid NUMBER, description
VARCHAR2(1000), ownerid NUMBER);
CREATE SEQUENCE lexeme_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE lexeme ADD CONSTRAINT pk_lexeme PRIMARY KEY (id);
ALTER TABLE lexeme ADD CONSTRAINT lexeme_conceptid_fk FOREIGN KEY
(conceptid) REFERENCES concept (id);
ALTER TABLE lexeme ADD CONSTRAINT lexeme_languageid_fk FOREIGN KEY
(languageid) REFERENCES language2 (id);
ALTER TABLE lexeme ADD CONSTRAINT lexeme_writingsystemid_fk FOREIGN KEY
(writingsystemid) REFERENCES writingsystem (id);
ALTER TABLE lexeme ADD CONSTRAINT lexeme_charcodesetid_fk FOREIGN KEY
(charcodesetid) REFERENCES charcodeset (id);
CREATE INDEX lexeme_conceptid ON lexeme(conceptid);
CREATE INDEX lexeme_languageid ON lexeme(languageid);
CREATE INDEX lexeme_writingsystemid ON lexeme(writingsystemid);
CREATE INDEX lexeme_charcodesetid ON lexeme(charcodesetid);
commit;
```

3.9. OBJECT_ID_SEQ

A nevek és a névhordozók közös azonosító-szekvenciája.

CREATE SEQUENCE object_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
commit;

3.10. CORPORATE

A testület típusú névhordozók táblája

CREATE TABLE corporate (id NUMBER NOT NULL, corporatetypeid NUMBER, address VARCHAR2(250), postaladdress VARCHAR2(250), phone VARCHAR2(250), fax VARCHAR2(250), email VARCHAR2(250), url VARCHAR2(250), duplumcheck VARCHAR2(20) DEFAULT '0', ownerid NUMBER);
ALTER TABLE corporate ADD CONSTRAINT pk_corporate PRIMARY KEY (id);
ALTER TABLE corporate ADD CONSTRAINT corporate_conceptid_fk FOREIGN KEY (corporatetypeid) REFERENCES concept (id);
CREATE INDEX corporate_corporatetypeid ON corporate(corporatetypeid);
CREATE INDEX corporate_ownerid ON corporate(ownerid);
commit;

3.11. CORPORATENAME

A testület típusú nevek táblája

CREATE TABLE corporatename (id NUMBER NOT NULL, ndaobjectid NUMBER, name VARCHAR2(250), shortname VARCHAR2(50), disambiguator VARCHAR2(250), pronunciation VARCHAR2(250), nametypeid NUMBER, languageid NUMBER, writingsystemid NUMBER, charcodesetid NUMBER, namelength NUMBER, ownerid NUMBER);
ALTER TABLE corporatename ADD CONSTRAINT pk_corporatename PRIMARY KEY (id);
ALTER TABLE corporatename ADD CONSTRAINT corporatename_ndaobjectid_fk FOREIGN KEY (ndaobjectid) REFERENCES corporate (id);
ALTER TABLE corporatename ADD CONSTRAINT corporatename_languageid_fk FOREIGN KEY (languageid) REFERENCES language2 (id);
ALTER TABLE corporatename ADD CONSTRAINT corporatename_writingsystemid_fk FOREIGN KEY (writingsystemid) REFERENCES writingsystem (id);
ALTER TABLE corporatename ADD CONSTRAINT corporatename_charcodesetid_fk FOREIGN KEY (charcodesetid) REFERENCES charcodeset (id);
ALTER TABLE corporatename ADD CONSTRAINT corporatename_nametypeid_fk FOREIGN KEY (nametypeid) REFERENCES concept (id);
CREATE INDEX corporatename_name ON corporatename(name);
CREATE INDEX corporatename_ndaobjectid ON corporatename(ndaobjectid);
CREATE INDEX corporatename_languageid ON corporatename(languageid);
CREATE INDEX corporatename_writingsystemid ON corporatename(writingsystemid);
CREATE INDEX corporatename_charcodesetid ON corporatename(charcodesetid);
CREATE INDEX corporatename_nametypeid ON corporatename(nametypeid);

CREATE INDEX corporatename_ownerid ON corporatename(ownerid);
commit;

3.12. CORPORATENAME_PREFERENCE

A testületnevek kontextusait tartalmazó tábla.

CREATE TABLE corporatename_preference (id NUMBER NOT NULL, ndanameid NUMBER, contextid NUMBER, ownerid NUMBER);
CREATE SEQUENCE corpname_preference_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE corporatename_preference ADD CONSTRAINT pk_corporatename_preference PRIMARY KEY (id);
ALTER TABLE corporatename_preference ADD CONSTRAINT cname_pref_ndanameid_fk FOREIGN KEY (ndanameid) REFERENCES corporatename (id);
ALTER TABLE corporatename_preference ADD CONSTRAINT cname_pref_contextid_fk FOREIGN KEY (contextid) REFERENCES concept (id);
CREATE INDEX cname_pref_ndanameid ON corporatename_preference(ndanameid);
CREATE INDEX cname_pref_contextid ON corporatename_preference(contextid);
commit;

3.13. GEOOBJECT2

A földrajzi típusú névhordozók táblája

CREATE TABLE geoobject2 (id NUMBER NOT NULL, projectiontypeid NUMBER, supergeotypeid NUMBER, geotypeid NUMBER, northboundlat VARCHAR2(20), southboundlat VARCHAR2(20), westboundlong VARCHAR2(20), eastboundlong VARCHAR2(20), duplumcheck VARCHAR2(20) DEFAULT '0', ownerid NUMBER);
ALTER TABLE geoobject2 ADD CONSTRAINT pk_geoobject2 PRIMARY KEY (id);
ALTER TABLE geoobject2 ADD CONSTRAINT geoobject2_projectiontypeid_fk FOREIGN KEY (projectiontypeid) REFERENCES concept (id);
ALTER TABLE geoobject2 ADD CONSTRAINT geoobject2_supergeotypeid_fk FOREIGN KEY (supergeotypeid) REFERENCES concept (id);
ALTER TABLE geoobject2 ADD CONSTRAINT geoobject2_geotypeid_fk FOREIGN KEY (geotypeid) REFERENCES concept (id);
CREATE INDEX geoobject2_projectiontypeid ON geoobject2(projectiontypeid);
CREATE INDEX geoobject2_supergeotypeid ON geoobject2(supergeotypeid);
CREATE INDEX geoobject2_geotypeid ON geoobject2(geotypeid);
CREATE INDEX geoobject2_ownerid ON geoobject2(ownerid);
commit;

3.14. GEOOBJECTNAME2

A földrajzi típusú nevek táblája

CREATE TABLE geoobjectname2 (id NUMBER NOT NULL, ndaobjectid NUMBER, name VARCHAR2(250), disambiguator VARCHAR2(250), pronunciation VARCHAR2(250),

nametypeid NUMBER, languageid NUMBER, writingsystemid NUMBER, charcodesetid NUMBER, namelength NUMBER, ownerid NUMBER);
ALTER TABLE geoobjectname2 ADD CONSTRAINT pk_geoobjectname2 PRIMARY KEY (id);
ALTER TABLE geoobjectname2 ADD CONSTRAINT geoobjectname2_ndaobjectid_fk FOREIGN KEY (ndaobjectid) REFERENCES corporate (id);
ALTER TABLE geoobjectname2 ADD CONSTRAINT geoobjectname2_languageid_fk FOREIGN KEY (languageid) REFERENCES language2 (id);
ALTER TABLE geoobjectname2 ADD CONSTRAINT geoobjectname2_writingsystemid_fk FOREIGN KEY (writingsystemid) REFERENCES writingsystem (id);
ALTER TABLE geoobjectname2 ADD CONSTRAINT geoobjectname2_charcodesetid_fk FOREIGN KEY (charcodesetid) REFERENCES charcodeset (id);
ALTER TABLE geoobjectname2 ADD CONSTRAINT geoobjectname2_nametypeid_fk FOREIGN KEY (nametypeid) REFERENCES concept (id);
CREATE INDEX geoobjectname2_ndaobjectid ON geoobjectname2(ndaobjectid);
CREATE INDEX geoobjectname2_languageid ON geoobjectname2(languageid);
CREATE INDEX geoobjectname2_writingsystemid ON geoobjectname2(writingsystemid);
CREATE INDEX geoobjectname2_charcodesetid ON geoobjectname2(charcodesetid);
CREATE INDEX geoobjectname2_nametypeid ON geoobjectname2(nametypeid);
CREATE INDEX geoobject2_ownerid ON geoobject2(ownerid);
commit;

3.15. GEONAME_PREFERENCE

A földrajzi nevek kontextusait tartalmazó tábla

CREATE TABLE geoname_preference (id NUMBER NOT NULL, ndanameid NUMBER, contextid NUMBER, ownerid NUMBER);
CREATE SEQUENCE geoname_preference_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE geoname_preference ADD CONSTRAINT pk_geoname_preference PRIMARY KEY (id);
ALTER TABLE geoname_preference ADD CONSTRAINT gname_pref_ndanameid_fk FOREIGN KEY (ndanameid) REFERENCES geoobjectname2 (id);
ALTER TABLE geoname_preference ADD CONSTRAINT gname_pref_contextid_fk FOREIGN KEY (contextid) REFERENCES concept (id);
CREATE INDEX gname_pref_geoobjectnameid ON geoname_preference(geoobjectnameid);
CREATE INDEX gname_pref_contextid ON geoname_preference(contextid);
commit;

3.16. FOOTPRINT2

A földrajzi objektumok footprint adatait tartalmazó tábla.

CREATE TABLE footprint (id NUMBER NOT NULL, latitude VARCHAR2(20), longitude VARCHAR2(20), nextpoint NUMBER, geoobjectid NUMBER);
CREATE SEQUENCE footprint_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE footprint ADD CONSTRAINT pk_footprint PRIMARY KEY (id);

```
ALTER TABLE footprint ADD CONSTRAINT footprint_nextpoint_fk FOREIGN KEY
(nextpoint) REFERENCES footprint (id);
ALTER TABLE footprint ADD CONSTRAINT footprint_geoobjectid_fk FOREIGN KEY
(geoobjectid) REFERENCES geoobject2 (id);
ALTER TABLE footprint ADD CONSTRAINT footprint_ndaobjectid_fk FOREIGN KEY
(ndaobjectid) REFERENCES geoobject2 (id);
CREATE INDEX footprint_latitude ON footprint(latitude);
CREATE INDEX footprint_longitude ON footprint(longitude);
CREATE INDEX footprint_nextpoint ON footprint(nextpoint);
CREATE INDEX footprint_geoobjectid ON footprint(geoobjectid);
commit;
```

3.17. PERSON2

A személy típusú névhordozók táblája.

```
CREATE TABLE person2 (id NUMBER NOT NULL, sexid NUMBER, duplumcheck
VARCHAR2(20) DEFAULT '0', ownerid NUMBER);
CREATE SEQUENCE person2_id_seq MINVALUE 1 START WITH 1 INCREMENT BY
1;
ALTER TABLE person2 ADD CONSTRAINT pk_person2 PRIMARY KEY (id);
ALTER TABLE person2 ADD CONSTRAINT person2_sexid_fk FOREIGN KEY (sexid)
REFERENCES concept (id);
CREATE INDEX person2_sexid ON person2(sexid);
CREATE INDEX person2_ownerid ON person2(ownerid);
commit;
```

3.18. PERSONNAME2

A személy típusú nevek táblája.

```
CREATE TABLE personname2 (id NUMBER NOT NULL, ndaobjectid NUMBER, name
VARCHAR2(250), disambiguator VARCHAR2(250), pronunciation VARCHAR2(250),
nametypeid NUMBER, namebuildingsruleid NUMBER, languageid NUMBER,
writingsystemid NUMBER, charcodesetid NUMBER, namepartcheck VARCHAR2(20),
namelength NUMBER, ownerid NUMBER);
ALTER TABLE personname2 ADD CONSTRAINT pk_personname2 PRIMARY KEY (id);
ALTER TABLE personname2 ADD CONSTRAINT personname2_ndaobjectid_fk
FOREIGN KEY (ndaobjectid) REFERENCES corporate (id);
ALTER TABLE personname2 ADD CONSTRAINT personname2_ownerid_fk FOREIGN
KEY (ownerid) REFERENCES users2 (id);
ALTER TABLE personname2 ADD CONSTRAINT personname2_languageid_fk FOREIGN
KEY (languageid) REFERENCES language2 (id);
ALTER TABLE personname2 ADD CONSTRAINT personname2_writingsystemid_fk
FOREIGN KEY (writingsystemid) REFERENCES writingsystem (id);
ALTER TABLE personname2 ADD CONSTRAINT personname2_charcodesetid_fk
FOREIGN KEY (charcodesetid) REFERENCES charcodeset (id);
ALTER TABLE personname2 ADD CONSTRAINT personname2_nametypeid_fk
FOREIGN KEY (nametypeid) REFERENCES concept (id);
```

```
ALTER TABLE personname2 ADD CONSTRAINT personname2_namebuildingsruleid_fk
FOREIGN KEY (namebuildingsruleid) REFERENCES concept (id);
CREATE INDEX personname2_ndaobjectid ON personname2(ndaobjectid);
CREATE INDEX personname2_languageid ON personname2(languageid);
CREATE INDEX personname2_writingsystemid ON personname2(writingsystemid);
CREATE INDEX personname2_charcodesetid ON personname2(charcodesetid);
CREATE INDEX personname2_nametypeid ON personname2(nametypeid);
CREATE INDEX personname2_namebuildingsruleid ON
personname2(namebuildingsruleid);
CREATE INDEX personname2_ownerid ON personname2(ownerid);
commit;
```

3.19. PERSONNAME_PREFERENCE

A személynevek kontextusait tartalmazó tábla.

```
CREATE TABLE personname_preference (id NUMBER NOT NULL, ndanameid
NUMBER, contextid NUMBER, ownerid NUMBER);
CREATE SEQUENCE personname_preference_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE personname_preference ADD CONSTRAINT pk_geoname_preference
PRIMARY KEY (id);
ALTER TABLE personname_preference ADD CONSTRAINT pname_pref_ndanameid_fk
FOREIGN KEY (ndanameid) REFERENCES geoobjectname2 (id);
ALTER TABLE personname_preference ADD CONSTRAINT pname_pref_contextid_fk
FOREIGN KEY (contextid) REFERENCES concept (id);
CREATE INDEX pname_pref_ndanameid ON personname_preference(ndanameid);
CREATE INDEX pname_pref_contextid ON personname_preference(contextid);
commit;
```

3.20. PERSONNAME_PROFESSION

A személynevekhez és a hozzájuk tartozó foglalkozások kapcsolatát leíró tábla.

```
CREATE TABLE personname_profession (id NUMBER NOT NULL, ndanameid
NUMBER, professionid NUMBER);
CREATE SEQUENCE personname_profession_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE personname_profession ADD CONSTRAINT pk_personname_profession
PRIMARY KEY (id);
ALTER TABLE personname_profession ADD CONSTRAINT pname_prof_ndanameid_fk
FOREIGN KEY (ndanameid) REFERENCES personname2 (id);
ALTER TABLE personname_profession ADD CONSTRAINT pname_prof_professionid_fk
FOREIGN KEY (professionid) REFERENCES concept (id);
CREATE INDEX pname_prof_ndanameid ON personname_profession(ndanameid);
CREATE INDEX pname_prof_professionid ON personname_profession(professionid);
commit;
```

3.21. PERSONNAME_PART

A személynevek elemekre bontott alakjait tartalmazó tábla.

CREATE TABLE personname_part (id NUMBER NOT NULL, namepart VARCHAR2(250), ndanameid NUMBER, nameparttypeid NUMBER, rank NUMBER);
CREATE SEQUENCE personname_part_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE personname_part ADD CONSTRAINT pk_personname_part PRIMARY KEY (id);
ALTER TABLE personname_part ADD CONSTRAINT personname_part_ndanameid_fk FOREIGN KEY (ndanameid) REFERENCES personname2 (id);
ALTER TABLE personname_part ADD CONSTRAINT personname_part_nameparttypeid_fk FOREIGN KEY (nameparttypeid) REFERENCES concept (id);
CREATE INDEX personname_part_ndanameid ON personname_part(ndanameid);
CREATE INDEX personname_part_nameparttypeid ON personname_part(nameparttypeid);
commmit;

3.22. NAMEBUILDINGRANK

A névelemek sorbarendezési szabályait tartalmazó tábla

CREATE TABLE namebuildingrank (id NUMBER NOT NULL, namebuildingsruleid NUMBER, nameparttypeid NUMBER, rank NUMBER);
CREATE SEQUENCE namebuildingrank_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE namebuildingrank ADD CONSTRAINT pk_namebuildingrank PRIMARY KEY (id);
ALTER TABLE namebuildingrank ADD CONSTRAINT namebuildingrank_namebuildingsruleid_fk FOREIGN KEY (namebuildingsruleid) REFERENCES concept (id);
ALTER TABLE namebuildingrank ADD CONSTRAINT namebuildingrank_nameparttypeid_fk FOREIGN KEY (nameparttypeid) REFERENCES concept (id);
CREATE INDEX namebuildingrank_namebuildingsruleid ON namebuildingrank(namebuildingsruleid);
CREATE INDEX namebuildingrank_nameparttypeid ON namebuildingrank(nameparttypeid);
commit;

3.23. PATTERNS

Az automatikus névelemekre bontást segítő tábla.

CREATE TABLE patterns (id NUMBER NOT NULL, pattern VARCHAR2(250), namebuildingruleid NUMBER, elementcount NUMBER, patterncount NUMBER);
CREATE SEQUENCE patterns_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE patterns ADD CONSTRAINT pk_patterns PRIMARY KEY (id);

```
ALTER TABLE patterns ADD CONSTRAINT patterns_namebuildingruleid_fk FOREIGN
KEY (namebuildingruleid) REFERENCES concept (id);
CREATE INDEX patterns_namebuildingruleid ON patterns(namebuildingruleid);
commit;
```

3.24. NOTE

A nevekhez, vagy a névhordozókhoz tartozó megjegyzések táblája.

```
CREATE TABLE note (id NUMBER NOT NULL, objecttypeid NUMBER, ndaobjectid
NUMBER, ndanameid NUMBER, notetypeid NUMBER, languageid NUMBER, note
VARCHAR2(1024), resourcenameuri VARCHAR2(1024), ownerid NUMBER, createdate
DATE);
CREATE SEQUENCE note_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE note ADD CONSTRAINT pk_note PRIMARY KEY (id);
ALTER TABLE note ADD CONSTRAINT note_objecttypeid_fk FOREIGN KEY
(objecttypeid) REFERENCES concept (id);
ALTER TABLE note ADD CONSTRAINT note_notetypeid_fk FOREIGN KEY (notetypeid)
REFERENCES concept (id);
CREATE INDEX note_objecttypeid ON note(objecttypeid);
CREATE INDEX note_notetypeid ON note(notetypeid);
CREATE INDEX note_ownerid ON note(ownerid);
CREATE INDEX note_note ON note(note);
CREATE INDEX note_resourcenameuri ON note(resourcenameuri);
commit;
```

3.25. EVENT2

Az eseményeket tartalmazó tábla.

```
CREATE TABLE event2 (id NUMBER NOT NULL, objecttypeid NUMBER, ndaobjectid
NUMBER, ndanameid NUMBER, eventtypeid NUMBER, originalDateString
VARCHAR2(100), begintime NUMBER, endtime NUMBER, dateConfidenceType
VARCHAR2(1), dateConfidenceValue NUMBER, ownerid NUMBER);
CREATE SEQUENCE event2_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE event2 ADD CONSTRAINT pk_event2 PRIMARY KEY (id);
ALTER TABLE event2 ADD CONSTRAINT event2_objecttypeid_fk FOREIGN KEY
(objecttypeid) REFERENCES concept (id);
ALTER TABLE event2 ADD CONSTRAINT event2_eventtypeid_fk FOREIGN KEY
(eventtypeid) REFERENCES concept (id);
CREATE INDEX event2_objecttypeid ON event2(objecttypeid);
CREATE INDEX event2_ndaobjectid ON event2(ndaobjectid);
CREATE INDEX event2_ndanameid ON event2(ndanameid);
CREATE INDEX event2_eventtypeid ON event2(eventtypeid);
CREATE INDEX event2_ownerid ON event2(ownerid);
CREATE INDEX event2_originalDateString ON event2(originalDateString);
CREATE INDEX event2_begintime ON event2(begintime);
CREATE INDEX event2_endtime ON event2(endtime);
commit;
```

3.26. EVENT_OBJECT_NAME

Az események és a nevek, ill. a névhordozók kapcsolatát tartalmazó tábla.

```
CREATE TABLE event_object_name (id NUMBER NOT NULL, objecttypeid NUMBER,
ndaobjectid NUMBER, ndanameid NUMBER, eventid NUMBER);
CREATE SEQUENCE event_object_name_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE event_object_name ADD CONSTRAINT pk_event_object_name
PRIMARY KEY (id);
ALTER TABLE event_object_name ADD CONSTRAINT event_object_name_eventid_fk
FOREIGN KEY (eventid) REFERENCES event2 (id);
CREATE INDEX event_object_name_ndaobjectid ON event_object_name(ndaobjectid);
CREATE INDEX event_object_name_ndanameid ON event_object_name(ndanameid);
CREATE INDEX event_object_name_eventid ON event_object_name(eventid);
commit;
```

3.27. OBJECT_OBJECT

A névhordozók közötti kapcsolatokat tartalmazó tábla.

CREATE TABLE object_object (id NUMBER NOT NULL, relationtypeid NUMBER, leftobjectid NUMBER, rightobjectid NUMBER, eventid NUMBER);
CREATE SEQUENCE object_object_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE object_object ADD CONSTRAINT pk_object_object PRIMARY KEY (id);
ALTER TABLE object_object ADD CONSTRAINT object_object_relationtypeid_fk FOREIGN KEY (relationtypeid) REFERENCES concept (id);
ALTER TABLE object_object ADD CONSTRAINT object_object_eventid_fk FOREIGN KEY (eventid) REFERENCES event2 (id);
CREATE INDEX object_object_relationtypeid ON object_object(relationtypeid);
CREATE INDEX object_object_leftobjectid ON object_object(leftobjectid);
CREATE INDEX object_object_rightobjectid ON object_object(rightobjectid);
commit;

3.28. RELATIONPARTTYPES2

A névhordozók közötti kapcsolatokat definiáló tábla.

CREATE TABLE relationparttypes2 (id NUMBER NOT NULL, relationtypeid NUMBER, leftobjecttypeid NUMBER, rightobjecttypeid NUMBER);
CREATE SEQUENCE relationparttypes2_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE relationparttypes2 ADD CONSTRAINT pk_relationparttypes2 PRIMARY KEY (id);
ALTER TABLE relationparttypes2 ADD CONSTRAINT relationparttypes2_relationtypeid_fk FOREIGN KEY (relationtypeid) REFERENCES concept (id);
ALTER TABLE relationparttypes2 ADD CONSTRAINT relationparttypes2_leftobjecttypeid_fk FOREIGN KEY (leftobjecttypeid) REFERENCES concept (id);
ALTER TABLE relationparttypes2 ADD CONSTRAINT relationparttypes2_rightobjecttypeid_fk FOREIGN KEY (rightobjecttypeid) REFERENCES concept (id);
CREATE INDEX relationparttypes2_relationtypeid ON relationparttypes2(relationtypeid);
CREATE INDEX relationparttypes2_leftobjecttypeid ON relationparttypes2(leftobjecttypeid);
CREATE INDEX relationparttypes2_rightobjecttypeid ON relationparttypes2(rightobjecttypeid);
commit;

3.29. BOUNDEDTYPES

A nevekhez, illetve a névhordozókhoz kapcsolódó adattípusokat leíró tábla.

```
CREATE TABLE boundedtypes (id NUMBER NOT NULL, object_or_name NUMBER,
objecttypeid NUMBER, categoryid NUMBER, technicalnames VARCHAR2(100), datatype
NUMBER, boundingtype NUMBER);
CREATE SEQUENCE boundedtypes_id_seq MINVALUE 1 START WITH 1 INCREMENT
BY 1;
ALTER TABLE boundedtypes ADD CONSTRAINT pk_boundedtypes PRIMARY KEY
(id);
ALTER TABLE boundedtypes ADD CONSTRAINT boundedtypes_objecttypeid_fk
FOREIGN KEY (objecttypeid) REFERENCES concept (id);
ALTER TABLE boundedtypes ADD CONSTRAINT boundedtypes_categoryid_fk
FOREIGN KEY (categoryid) REFERENCES concept (id);
CREATE INDEX boundedtypes_object_or_name ON boundedtypes(object_or_name);
CREATE INDEX boundedtypes_objecttypeid ON boundedtypes(objecttypeid);
CREATE INDEX boundedtypes_categoryid ON boundedtypes(categoryid);
CREATE INDEX boundedtypes_datatype ON boundedtypes(datatype);
CREATE INDEX boundedtypes_boundingtype ON boundedtypes(boundingtype);
commit;
```

3.30. BOUNDEDTYPES FELTÖLTÉSE

A boundedtypes tábla feltöltése alapadatokkal. Ezt csak a concept, illetve a lexeme tábla feltöltése után szabad elvégezni!

```
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames,
datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '0', (SELECT conceptid
FROM lexeme WHERE name='személy'), (SELECT conceptid FROM lexeme WHERE
name='nem'), 'person2:sexid', '1', '1');
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames,
datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '0', (SELECT conceptid
FROM lexeme WHERE name='földrajzi'), (SELECT conceptid FROM lexeme WHERE
name='geotípus'), 'geoobject2:geotypeid', '0', '1');
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames,
datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '0', (SELECT conceptid
FROM lexeme WHERE name='földrajzi'), (SELECT conceptid FROM lexeme WHERE
name='szupergeotípus'), 'geoobject2:supergeotypeid', '0', '1');
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames,
datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '0', (SELECT conceptid
FROM lexeme WHERE name='testület'), (SELECT conceptid FROM lexeme WHERE
name='testülettípus'), 'corporate:corporatetypeid', '0', '1');
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames,
datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '1', (SELECT conceptid
FROM lexeme WHERE name='személy'), (SELECT conceptid FROM lexeme WHERE
name='névtípus'), 'personname2:nametypeid', '1', '1');
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames,
datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '1', (SELECT conceptid
FROM lexeme WHERE name='földrajzi'), (SELECT conceptid FROM lexeme WHERE
name='névtípus'), 'geoobjectname2:nametypeid', '1', '1');
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames,
datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '1', (SELECT conceptid
```

FROM lexeme WHERE name='testület'), (SELECT conceptid FROM lexeme WHERE name='névtípus'), 'corporatename:nametypeid', '1', '1');
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames, datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '1', (SELECT conceptid FROM lexeme WHERE name='személy'), (SELECT conceptid FROM lexeme WHERE name='foglalkozás'), 'personname_profession:professionid', '1', '0');
INSERT INTO boundedtypes (id, object_or_name, objecttypeid, categoryid, technicalnames, datatype, boundingtype) VALUES (boundedtypes_id_seq.nextval, '1', (SELECT conceptid FROM lexeme WHERE name='testület'), (SELECT conceptid FROM lexeme WHERE name='rövidnév'), 'corporatename:shortname', '0', '1');

3.31. HISTORY

A neveken, illetve a névhordozókon végzett módosításokat tartalmazó tábla.

CREATE TABLE history (id NUMBER NOT NULL, objecttypeid NUMBER, ndaobjectid NUMBER, ndanameid NUMBER, modrequest NVARCHAR2(2000), moddate DATE, ownerid NUMBER);
CREATE SEQUENCE history_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE history ADD CONSTRAINT pk_history PRIMARY KEY (id);
ALTER TABLE history ADD CONSTRAINT history_objecttypeid_fk FOREIGN KEY (objecttypeid) REFERENCES concept (id);
CREATE INDEX history_objecttypeid ON history(objecttypeid);
CREATE INDEX history_ndaobjectid ON history(ndaobjectid);
CREATE INDEX history_ndanameid ON history(ndanameid);
CREATE INDEX history_moddate ON history(moddate);
commit;

3.32. MODIFICATIONS

A nevekhez, illetve a névhordozókhoz tartozó módosítási javaslatokat tartalmazó tábla.

CREATE TABLE modifications (id NUMBER NOT NULL, objecttypeid NUMBER, ndaobjectid NUMBER, ndanameid NUMBER, modrequest CLOB, moddate DATE, ownerid NUMBER, request_user_id NUMBER);
CREATE SEQUENCE modifications_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE modifications ADD CONSTRAINT pk_modifications PRIMARY KEY (id);
ALTER TABLE modifications ADD CONSTRAINT modifications_objecttypeid_fk FOREIGN KEY (objecttypeid) REFERENCES concept (id);
CREATE INDEX modifications_objecttypeid ON modifications(objecttypeid);
CREATE INDEX modifications_ndaobjectid ON modifications(ndaobjectid);
CREATE INDEX modifications_ndanameid ON modifications(ndanameid);
CREATE INDEX modifications_moddate ON modifications(moddate);
CREATE INDEX modifications_request_user_id ON modifications(request_user_id);
commit;

3.33. TRANSFERREDNAMES

Az összevont nevek elérését segítő tábla.

```
CREATE TABLE transferrednames (id NUMBER NOT NULL, oldid NUMBER, newid
NUMBER);
CREATE SEQUENCE transferrednames_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE transferrednames ADD CONSTRAINT pk_transferrednames PRIMARY
KEY (id);
ALTER TABLE transferrednames ADD CONSTRAINT transferrednames_unique
UNIQUE(oldid, newid);
CREATE INDEX transferrednames_oldid ON transferrednames(oldid);
CREATE INDEX transferrednames_newid ON transferrednames(newid);
commit;
```

3.34. LOG_DUPLUMS

A duplumok statisztikáját tartalmazó tábla.

```
CREATE TABLE log_duplums (id NUMBER NOT NULL, duplumdate DATE, surecount
NUMBER, possiblecount NUMBER);
CREATE SEQUENCE log_duplums_id_seq MINVALUE 1 START WITH 1 INCREMENT
BY 1;
ALTER TABLE log_duplums ADD CONSTRAINT pk_log_duplums PRIMARY KEY (id);
CREATE INDEX log_duplums_duplumdate ON log_duplums(duplumdate);
commit;
```

3.35. LOG_DUPLUMSFORUSERS

A duplumok statisztikáját adatgazdákra lebontva tartalmazó tábla.

```
CREATE TABLE log_duplumsforusers (id NUMBER NOT NULL, userid NUMBER,
surecount NUMBER, possiblecount NUMBER);
CREATE SEQUENCE log_duplumsforusers_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_duplumsforusers ADD CONSTRAINT pk_log_duplumsforusers
PRIMARY KEY (id);
CREATE INDEX log_duplumsforusers_userid ON log_duplumsforusers(userid);
commit;
```

3.36. LOG_LOGIN

A bejelentkezések statisztikáját tartalmazó tábla.

```
CREATE TABLE log_login (id NUMBER, userid NUMBER, logindate DATE);
CREATE SEQUENCE log_login_id_seq MINVALUE 1 START WITH 1 INCREMENT BY
1;
ALTER TABLE log_login ADD CONSTRAINT pk_log_login PRIMARY KEY (id);
CREATE INDEX log_login_userid ON log_login(userid);
```

```
CREATE INDEX log_login_logindate ON log_login(logindate);
commit;
```

3.37. LOG_MERGEDRECORDS

Az összevont rekordok statisztikáját tartalmazó tábla.

```
CREATE TABLE log_mergedrecords (id NUMBER NOT NULL, recorddate DATE, userid
NUMBER, recordcount NUMBER);
CREATE SEQUENCE log_mergedrecords_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_mergedrecords ADD CONSTRAINT pk_log_mergedrecords
PRIMARY KEY (id);
CREATE INDEX log_mergedrecords_userid ON log_mergedrecords(userid);
CREATE INDEX log_mergedrecords_logindate ON log_mergedrecords(recorddate);
commit;
```

3.38. LOG_MODIFIEDRECORDS

Az módosított rekordok statisztikáját tartalmazó tábla.

```
CREATE TABLE log_modifiedrecords (id NUMBER NOT NULL, objecttypeid NUMBER,
newcount NUMBER, modcount NUMBER, countdate DATE);
CREATE SEQUENCE log_modifiedrecords_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_modifiedrecords ADD CONSTRAINT pk_log_modifiedrecords
PRIMARY KEY (id);
CREATE INDEX log_modifiedrecords_countdate ON log_modifiedrecords(countdate);
commit;
```

3.39. LOG_MODIFIEDRECORDS_TEMP

Az összevont rekordok statisztikájához szükséges segédtábla.

```
CREATE TABLE log_modifiedrecords_temp (id NUMBER NOT NULL, ndanameid
NUMBER, moddate DATE);
CREATE SEQUENCE log_modifiedrecords_temp_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_modifiedrecords_temp ADD CONSTRAINT
pk_log_modifiedrecords_temp PRIMARY KEY (id);
CREATE INDEX log_modifiedrecords_temp_ndanameid ON
log_modifiedrecords_temp(ndanameid);
CREATE INDEX log_modifiedrecords_temp_moddate ON
log_modifiedrecords_temp(moddate);
```

3.40. LOG_MOSTREFERREDNAMES

A legtöbbet referált nevek statisztikájához szükséges tábla.

```
CREATE TABLE log_mostreferrednames (id NUMBER NOT NULL, topdate DATE, t1
VARCHAR2(500), t2 VARCHAR2(500), t3 VARCHAR2(500), t4 VARCHAR2(500), t5
VARCHAR2(500));
CREATE SEQUENCE log_mostreferrednames_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_mostreferrednames ADD CONSTRAINT pk_log_mostreferrednames
PRIMARY KEY (id);
CREATE INDEX log_mostreferrednames_topdate ON log_mostreferrednames(topdate);
commit;
```

3.41. LOG_MOSTSEARCHEDNAMES

A legtöbbet keresett kifejezések statisztikájához szükséges tábla.

```
CREATE TABLE log_mostsearchednames (id NUMBER NOT NULL, topdate DATE, t1
VARCHAR2(500), t2 VARCHAR2(500), t3 VARCHAR2(500), t4 VARCHAR2(500), t5
VARCHAR2(500));
CREATE SEQUENCE log_mostsearchednames_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_mostsearchednames ADD CONSTRAINT pk_log_mostsearchednames
PRIMARY KEY (id);
CREATE INDEX log_mostsearchednames_topdate ON log_mostsearchednames(topdate);
commit;
```

3.42. LOG_NAMECOUNT

Az adatgazánkénti rekordok számának statisztikájához szükséges tábla.

```
CREATE TABLE log_namecount (id NUMBER NOT NULL, userid NUMBER,
objecttypeid NUMBER, namecount NUMBER, countdate DATE);
CREATE SEQUENCE log_namecount_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_namecount ADD CONSTRAINT pk_log_namecount PRIMARY KEY
(id);
CREATE INDEX log_namecount_userid ON log_namecount(userid);
CREATE INDEX log_namecount_objecttypeid ON log_namecount(objecttypeid);
commit;
```

3.43. LOG_NAMERECORDS

A névterenkénti rekordok számának statisztikájához szükséges tábla.

```
CREATE TABLE log_namerecords (id NUMBER NOT NULL, objecttypeid NUMBER,
nameholdercount NUMBER, namecount NUMBER, recorddate DATE);
CREATE SEQUENCE log_namerecords_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_namerecords ADD CONSTRAINT pk_log_namerecords PRIMARY
KEY (id);
CREATE INDEX log_namerecords_objecttypeid ON log_namerecords(objecttypeid);
```

CREATE INDEX log_namerecords_recorddate ON log_namerecords(recorddate);
commit;

3.44. LOG_NAMESPACEUSAGE

A leggyakoribb névtértípus statisztikájához szükséges tábla.

CREATE TABLE log_namespaceusage (id NUMBER NOT NULL, personusage NUMBER, geousage NUMBER, corporateusage NUMBER, recorddate DATE);
CREATE SEQUENCE log_namespaceusage_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE log_namespaceusage ADD CONSTRAINT pk_log_namespaceusage PRIMARY KEY (id);
CREATE INDEX log_namespaceusage_recorddate ON log_namespaceusage(recorddate);
commit;

3.45. LOG_SEARCH

A keresések statisztikájához szükséges tábla.

CREATE TABLE log_search (id NUMBER NOT NULL, searchexp VARCHAR2(500), userid NUMBER, searchcount NUMBER, objecttypeid NUMBER, lastsearch DATE);
CREATE SEQUENCE log_search_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE log_search ADD CONSTRAINT pk_log_search PRIMARY KEY (id);
CREATE INDEX log_search_searchexp ON log_search(searchexp);
CREATE INDEX log_search_userid ON log_search(userid);
CREATE INDEX log_search_lastsearch ON log_search(lastsearch);
CREATE INDEX log_search_objecttypeid ON log_search(objecttypeid);
commit;

3.46. LOG_SEARCHCOUNT

A keresések számának statisztikájához szükséges tábla.

CREATE TABLE log_searchcount (id NUMBER, searchdate DATE, usercount NUMBER, guestcount NUMBER);
CREATE SEQUENCE log_searchcount_id_seq MINVALUE 1 START WITH 1 INCREMENT BY 1;
ALTER TABLE log_searchcount ADD CONSTRAINT pk_log_searchcount PRIMARY KEY (id);
CREATE INDEX log_searchcount_searchdate ON log_searchcount(searchdate);
commit;

3.47. LOG_SHOWDATA

Az adatmegjelenítések számának statisztikájához szükséges tábla.

CREATE TABLE log_showdata (id NUMBER, showdate DATE, showcount NUMBER);

```
CREATE SEQUENCE log_showdata_id_seq MINVALUE 1 START WITH 1 INCREMENT
BY 1;
ALTER TABLE log_showdata ADD CONSTRAINT pk_log_showdata PRIMARY KEY
(id);
CREATE INDEX log_showdata_showdate ON log_showdata(showdate);
commit;
```

3.48. LOG_TOPUSERS

A legtöbb rekorddal rendelkező adatgazdák statisztikájához szükséges tábla.

```
CREATE TABLE log_topusers (id NUMBER NOT NULL, topdate DATE, t1 NUMBER, t2
NUMBER, t3 NUMBER, t4 NUMBER, t5 NUMBER);
CREATE SEQUENCE log_topusers_id_seq MINVALUE 1 START WITH 1 INCREMENT
BY 1;
ALTER TABLE log_topusers ADD CONSTRAINT pk_log_topusers PRIMARY KEY (id);
CREATE INDEX log_topusers_topdate ON log_topusers(topdate);
commit;
```

3.49. LOG_WEBSITECLICK

Az adatgazda intézményeinek weboldalaira való hivatkozások statisztikájához szükséges tábla.

```
CREATE TABLE log_websiteclick (id NUMBER NOT NULL, userid NUMBER, clickcount
NUMBER);
CREATE SEQUENCE log_websiteclick_id_seq MINVALUE 1 START WITH 1
INCREMENT BY 1;
ALTER TABLE log_websiteclick ADD CONSTRAINT pk_log_websiteclick PRIMARY
KEY (id);
CREATE INDEX log_websiteclick_userid ON log_websiteclick(userid);
commit;
```

4. NDA-protokoll

Az NDA-protokoll külső rendszerből történő alkalmazása úgy történik, hogy az alkalmazó program meghívja a https://ndans1.nda.hu/keresfeldolgozo.php fájlt, úgy hogy az XML kérést POST, vagy GET metódussal elküldi. A visszaérkező karaktersorozat maga a válasz XML, amelyet az alkalmazó programnak kell feldolgoznia. Az NDA-protokoll kéréseiről és válaszairól, azok XML szerkezetéről bővebb információ az NDA tulajdonnévtér rendszer fizikai tervében található.