Debreceni Egyetem
Informatikai Kar

Egy civil szervezet eszköz- és tagnyilvántartása

Témavezető:
Kósa Márk
egyetemi tanársegéd

Készítette:
Csillag Péter
programtervező informatikus (Bsc)

DEBRECEN
2009

Tartalomjegyzék

I. Bevezetés

Az Egri Vitézlő Oskola Katonai Hagyományőrző és Sport Közhasznú Egyesület 1996 őszén alakult Eger történelmi hagyományaihoz kapcsolódva. A tagok az egykor volt törökkori végvári hagyományok felélesztését tűzték ki célul. A szervezet fő tevékenységi köre a végvári vitézek harcművészetének és szellemiségének ápolása, illetve bemutatása a nagyközönség előtt. Előadásaikon életképeket mutatnak be a török hódoltság korából, íjászattal, gyalogos párviadalokkal, pirotechnikai elemekkel fűszerezve. Mindezek mellett egyfajta önképző kör gyanánt történeti témájú ismeretterjesztő előadások, illetve tanulmányi kirándulások szervezésével kínál ismeretszerzési lehetőséget a csatlakozó fiatalok számára.

Az egyesületnek az egri Dobó István Vármúzeum ad otthont, mellyel 1998 őszétől szervezetileg is összekapcsolódott. Ennek eredményeként napjainkban már a Vitézlő Oskola, mint az Eger Vára Barátainak Köre ifjúsági tagozata működik.

Az egyesület a produkciók kidolgozásánál nagy hangsúlyt fektet a látványra, a korhűségre és a biztonságra. Ennek érdekében a megfelelő ismeretek és készségek elsajátítása heti rendszerességgel elméleti, valamint - szakedző irányításával - gyakorlati (vívó, íjász) foglalkozások keretében történnek. A fellépéseken használt ruhák és fegyverek eredeti darabok alapján készült pontos másolatok.

A tagság növekvő létszáma, valamint az előadások során egyre több használatba kerülő fegyver és egyéb eszköz miatt a szervezetnek mára szükségessé vált egy új, megbízható nyilvántartó rendszer kifejlesztése. Ennek segítségével mind a tagság, mind az egyesület által használt eszközöket könnyedén és átláthatóan nyilván lehet tartani.

II. Áttekintés dokumentum

Általános leírás

Az Abydos egy online, az Egri Vitézlő Oskola által használt számítógépes rendszer, amely lehetőséget biztosít arra, hogy az egyesület tagjai, valamint más, a szervezet iránt érdeklődők információkat kaphassanak a csoportról. A weben keresztül elérhető program segítségével a felhasználók tájékozódhatnak a tagsággal és az eszközökkel kapcsolatos dolgokról, és bizonyos felhasználók módosíthatják is az adatbázis egy-egy részét. A szoftver ezen kívül megvalósítja az Oskola tagjainak és az általuk használt ruhák, fegyverek, és egyéb eszközök egyszerű és bárkinek könnyen hozzáférhető nyilvántartását. A rendszer webes felületén keresztül az érdeklődőknek lehetőségük van a tagok, és az eszközök közti válogatásra, különböző szempontok szerinti keresésre, vagy felhasználótól függően az adatok módosítására is. Az egyesület tagjait az adminisztrátor a program üzembe helyezésekor regisztrálja, így a tagoknak csupán be kell jelentkezni a kapott jelszó segítségével. (A szervezethez tagsági viszonnyal nem kötődő felhasználók bejelentkezés nélkül használhatják a rendszer szolgáltatásait, ám az ő esetükben sokkal kevesebb információ érhető el.) Bejelentkezett felhasználók az információszerzés mellett igénybe vehetik a program módosítási szolgáltatásait is, mely segítségével átírhatják az adatbázis bizonyos részeit.

Az Abydos rendszernek elsődleges célja tehát a Vitézlő Oskola által használt eszközöknek, valamint a regisztrált tagoknak a szakszerű nyilvántartása. Mindemellett másodlagos célkitűzés, hogy a vezetőség, valamint a tagság naprakész információkhoz juthassanak az egyesületről és a tagság által használt felszerelésekről, így biztosítva könnyebb átláthatóságot és egyszerűbb eszközbeszerzési lehetőségeket.

A szoftvert háromféle személy használhatja. Az általános felhasználó, vagy vendég, akinek nem kell bejelentkeznie a rendszer használatához. Az egyesület tagja, akinek már be kell jelentkezni, viszont ő már módosíthatja is a saját adatait. Továbbá a rendszergazda, vagy más néven adminisztrátor, aki teljes joggal rendelkezik a szervezet nyilvántartását illetően. Törölhet, vagy vehet fel új tagokat illetve eszközöket, valamint módosíthatja is az azokhoz tartozó adatokat.

Általános követelmények

- A kivitelező köteles minden üzleti logikához és felületi elemhez tartozó döntésükről kikérni az Egri Vitézlő Oskola véleményét, hozzájárulását. A fejlesztés minden lényeges pontja, a fejlesztési eredmények, és a fejlesztés teljes dokumentációja elérhető kell, hogy legyen a megrendelő számára. A fejlesztés alatt, és annak végeztével a forráskód és a teljes dokumentáció tulajdonjoga a megrendelőt illeti meg.
- A szoftverfejlesztés minden lépése az *ISO IEC 90003 2004 Software Standard*-nak megfelelően kell történnie, melynek következetes betartásából származó dokumentumok a megrendelő számára hozzáférhetőek kell, hogy legyenek a fejlesztés bármely fázisa alatt. A dokumentálás „OpenDocument Text” (ISO/IEC 26300:2006) formátumban kell történnie.
- A felhasználók által használt kezelőfelület legyen egyszerű, áttekinthető, ezzel is biztosítva a könnyebb kezelhetőséget. Mivel a program az egyesület honlapjáról lesz elérhető, fontos kritérium az is, hogy a megjelenési stílus ne, vagy csupán minimálisan térjen el az ott használatostól. Ezért a szükséges erőforrásokat a megrendelő biztosítja a kivitelezőnek.
- A rendszer minden művelet eredményéről tájékoztatja a szoftver használóját. Hiba esetén értesítést küld a probléma fajtájától függően az aktuális felhasználónak, vagy a rendszeradminisztrátornak.
- A program minden elvégzett és végrehajtott műveletről (függetlenül attól, hogy sikeres, vagy sikertelen) naplózást kell, hogy végezzen. Naplózásra kell, hogy kerüljön a teljes adatbázis és a szoftver működésének eseményei, hibái. A naplózás szerkezetében el kell különülnie a különböző területekről érkező eseményeknek. Egy-egy naplóbejegyzésnek teljesen egyértelműnek és átláthatónak kell lennie. Naplózásra kell kerülnie a rendszer felhasználóinak a belépése és kilépése is.
- A regisztrált felhasználók jelszavait külön, bizalmas információként kell tárolni.
- A rendszer figyeli, hogy a tagok felhasználói nevei egyediek legyenek, így nem fordulhat elő két egyforma.

Rendszerkövetelmények

- Operációs rendszer: platform függetlennek kell lennie
- Programozási nyelv: Java technológia
- Célhardver: A szoftver igényeihez fognak igazodni, a tervezéskor nem kell figyelembe venni
- Várható felhasználók száma: 50 fő

III. Fogalomszótár

Felhasználó az Abydos-t használó személy

Vendég olyan felhasználó, aki jogilag nem tag az egyesületnél, a rendszer használatához regisztráció és bejelentkezés nem szükséges

Tag az egyesülettel tagsági viszonyban álló személy, regisztráció után bejelentkezés jelszóval lehetséges

Adminisztrátor, rendszergazda az elnökség által kijelölt tag

Eszköz, felszerelés az Oskola tulajdonában lévő eszközök

Ruha a fellépéseken használt ruhák (felszerelés)

Fegyver a fellépéseken használt fegyverek (felszerelés)

Regisztráció az aktuális tag szükséges adatainak megadása, melynek eredménye képen a felhasználó email-ben megkapja a jelszavát

Tag státusza a tag egyesületben betöltött státusza (elnökségi tag, rendes tag, próbaidős tag, pártoló tag, adminisztrátor)

Eszköz tulajdonos az aktuális eszköz melyik taghoz van kiosztva (minden, a nyilvántartásba szereplő eszköz a Vitézlő Oskola tulajdona)

Eszköz típus, kategória az aktuális eszköz milyen típusba van besorolva (fegyver, ruha, védő felszerelés, kellék)

IV. Szakterületi kapcsolatok és folyamatok

Regisztrálás az aktuális tag szükséges adatainak megadása, melynek eredménye képen a felhasználó email-ben megkapja a jelszavát

Bejelentkezés az aktuális tag megadja a felhasználói nevét és a jelszavát

Lekérdezés a felhasználó információkat gyűjt az adatbázisból

Módosítás a tag átírhatja az adatbázis bizonyos értékeit

Keresés a felhasználó adatokat keres az nyilvántartásban, kategória, vagy tárgyszó alapján

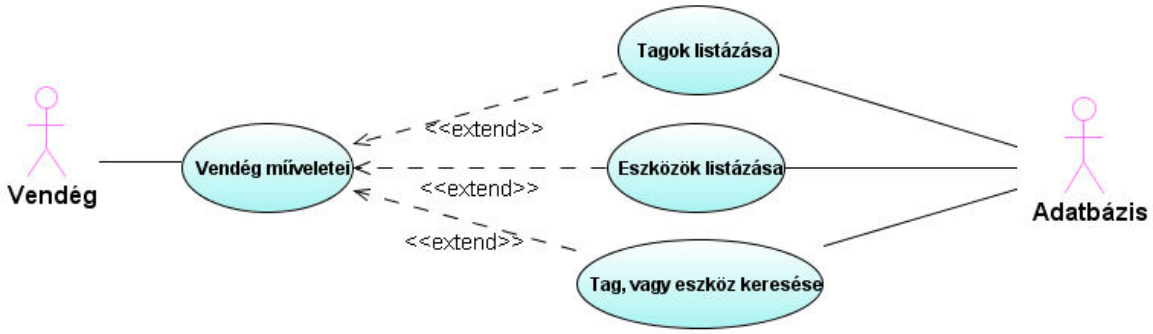
V. Forgatókönyv

Általános felhasználók számára

Az Abydos egy teljesen nyilvános rendszer, ugyanis weben keresztül bárki elérheti magát a rendszert, és annak bizonyos szolgáltatásait. Induláskor egy üdvözlő kép lesz látható, ahol a felhasználó tájékozódhat, hogy mi a szoftver célja, milyen szolgáltatásokat nyújt, valamint hogy milyen elvárásai vannak a programnak a felhasználó felé (például bejelentkezés).

Egy általános felhasználó adatai nincsenek nyilvántartva a rendszerben, így a vendégnek nem szükséges (és nem is lehetséges) regisztrálnia, vagy bejelentkeznie. Egy felhasználónak tehát rögtön a kezdő oldalról lehetősége van belépni a rendszerbe, mint vendég. Ezt követően a következő funkciókat jogosult használni:

1. Tagok listázása
2. Eszközök listázása
3. Adott kategóriába, vagy adott tulajdonoshoz tartozó eszközök listázása
4. Tárgyszó alapján eszközök, vagy tagok keresése



1. ábra – Vendég használati eset (use case) diagram

Eszközt, vagy tagot keresni úgy lehet, hogy a felhasználó megadja az általa ismert adatokat, majd a rendszer kilistázza az ezen adatoknak megfelelő bejegyzéseket. Tagok listázásánál, vagy keresésénél egy-egy bejegyzés tartalmazza az adott tag nevét, státuszát, belépési dátumát, valamint a fényképét, míg eszközök listázásánál, vagy keresésénél az adott eszköz nevét, típusát, tulajdonosát, beszerzésének idejét, és a hozzá tartozó képet.

Az általános, vagy vendég felhasználó tehát kizárólag lekérdezéseket hajthat végre, azaz a nyilvántartás aktuális tartalmát tekintheti meg, ám mindezt korlátozott módon, mivel értelemszerűen nem láthatja például a tagság személyes adatait. Az általános felhasználó módosítási joggal nem rendelkezik.

Az egyesület tagjai számára

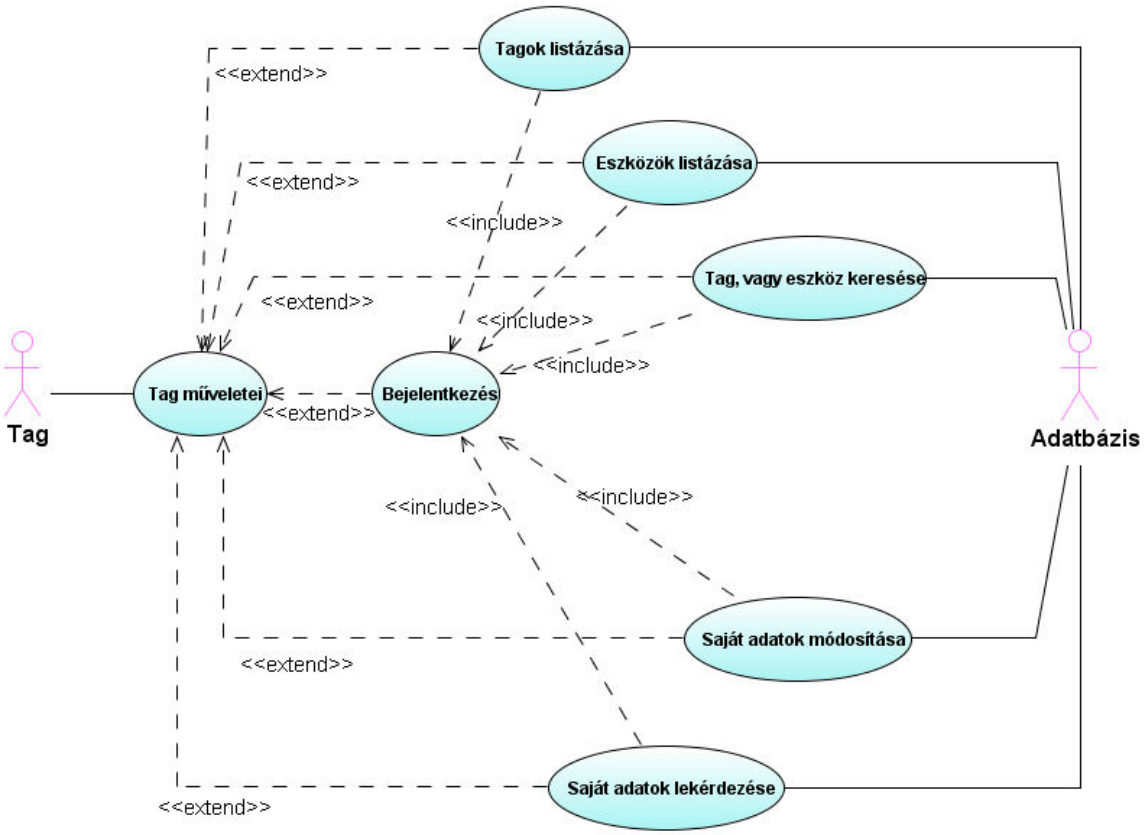
Az Abydos rendszert webes felületen keresztül érheti el a felhasználó. A program használatának kezdetekor egy üdvözlő kép lesz látható, amely egy rövid leírást ad a szoftverről, annak céljáról, valamint hogy milyen szolgáltatásokat nyújt, és hogy milyen elvárásai vannak a programnak a felhasználó felé (például bejelentkezés).

Az Egri Vitézlő Oskola tagjai – optimális esetben – szerepelnek a nyilvántartásban. Ahhoz, hogy egy tag igénybe vehesse a rendszer által nyújtott szolgáltatásokat, be kell jelentkeznie. A felhasználónak rögtön a kezdő oldalról lehetősége van bejelentkezni a rendszerbe, ahol meg kell adnia a korábban meghatározott felhasználói nevet, és jelszót.

Amennyiben a felhasználó még nem rendelkezik ilyennel, úgy regisztráltatnia kell, hogy megkapja ezeket. A regisztrációt maga a felhasználó nem végezheti el, hanem egy ebben illetékes adminisztrátor segítségét kell kérnie, aki elvégzi a szükséges teendőket. A regisztrációhoz meg kell adni néhány személyes adat mellett egy e-mail címet, ahová a regisztrálás után a rendszer el fogja küldeni a felhasználó leendő felhasználói nevét, és jelszavát. A megadott adatokat a rendszer külön, illetéktelen személyek hozzáférésétől elzártan tartja nyilván.

A tagok a bejelentkezés után a következő szolgáltatásokat vehetik igénybe:

1. Tagok listázása
2. Eszközök listázása
3. Adott kategóriába, vagy adott tulajdonoshoz tartozó eszközök listázása
4. Tárgyszó alapján eszközök, vagy tagok keresése
5. Saját adatok lekérdezése
6. Saját felhasználói név, jelszó, e-mail cím, lakcím, vagy telefonszám módosítása



2. ábra – Tag használati eset (use case) diagram

A bejelentkezést követően tehát az egyesület tagja igénybe veheti a különféle szolgáltatásokat. Egy tag jogosultságát tekintve rendelkezik mindazzal, amivel egy általános felhasználó, annyi eltéréssel, hogy egy-egy lekérdezés eredménye több információval szolgál az aktuális tagról, illetve eszközről. A felhasználónak lehetősége nyílik a saját adatainak lekérdezésére, valamint a saját adatainak a módosítására is, habár ez a funkció csak korlátozott mértékben érhető el a tagok számára, hiszen például a nem változó adatok (születési hely, születési idő, leánykori név, anyja neve, belépés dátuma) megváltoztatására nincs lehetősége.

Tagok listázásánál az aktuálisan a rendszert használó tag látja mindazokat az adatokat, amiket egy vendég felhasználó is, viszont ezen kívül egy-egy bejegyzésben megjelenik a lakcím, az e-mail cím, valamint a telefonszám is. Az eszközök listázásánál hasonlóképp plusz információként jelenik meg az eszközhöz tartozó megjegyzés.

Adminisztrátorok számára

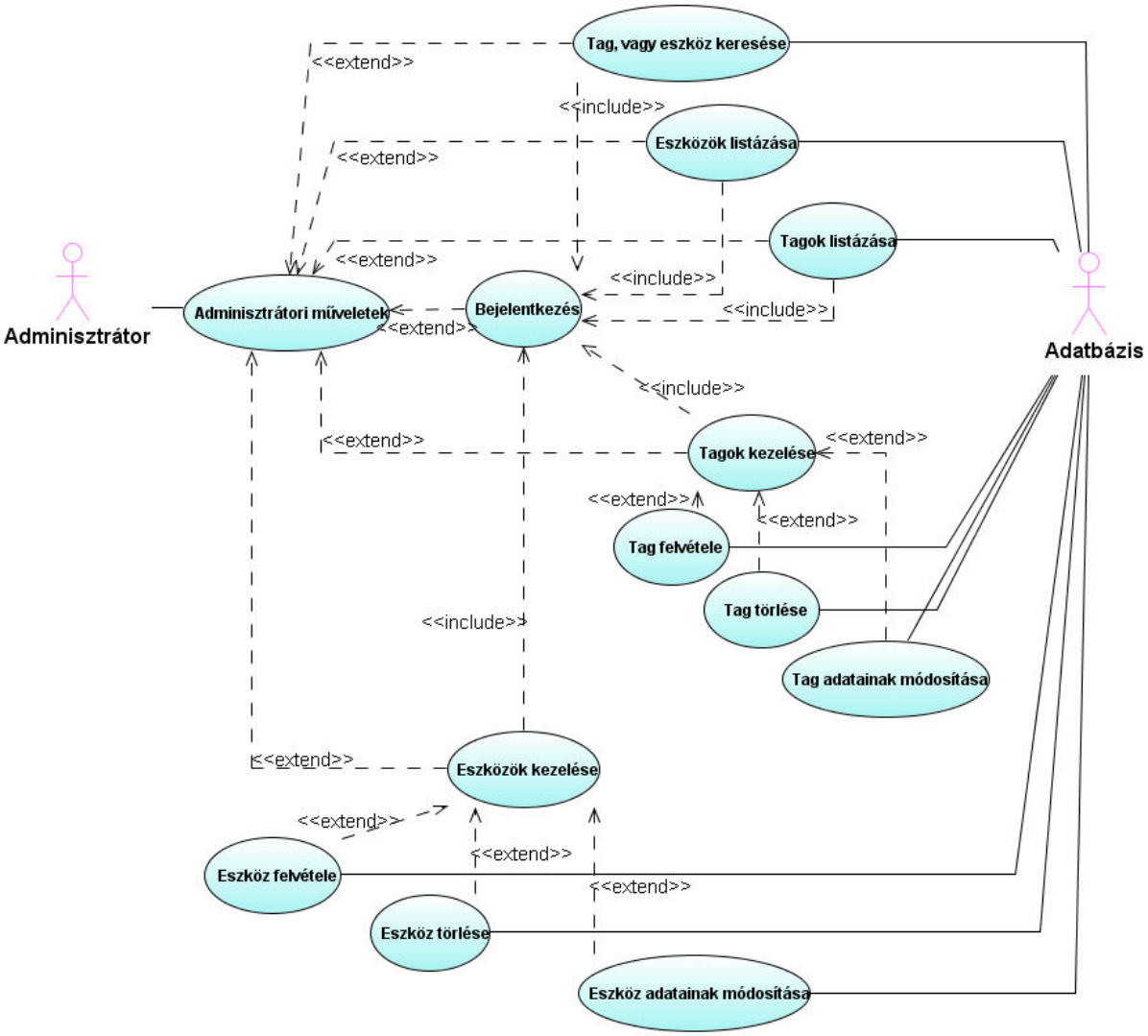
Az adminisztrátor, más felhasználókhoz hasonlóan interneten keresztül érheti el a rendszert, melynek üdvözlő képernyőjén információkat kaphat a szoftverről. Tájékozódhat, hogy mi a program célja, milyen szolgáltatásai vannak, valamint hogy mit kell tennie a felhasználónak ahhoz, hogy a rendszer által nyújtott szolgáltatásokat igénybe tudja venni (például bejelentkezés).

A rendszer adminisztrátorai – a korábban leírtak szerint – az egyesület tagjai is egyben, így emiatt ők is szerepelnek a nyilvántartásban. Ahhoz, hogy egy rendszergazda igénybe vehesse a rendszer által nyújtott szolgáltatásokat, neki is be kell jelentkeznie. A felhasználónak rögtön a kezdő oldalról lehetősége van bejelentkezni a rendszerbe, ahol meg kell adnia a korábban meghatározott felhasználói nevet, és jelszót.

Az adminisztrátorok a következő szolgáltatásokat jogosultak használni:

1. Tagok listázása
2. Eszközök listázása
3. Adott kategóriába, vagy adott tulajdonoshoz tartozó eszközök listázása
4. Tárgyszó alapján eszközök, vagy tagok keresése
5. Tag felvétele

6. Tag törlése
7. Tag adatainak módosítása
8. Eszköz felvétele
9. Eszköz törlése
10. Eszköz adatainak módosítása



3. ábra – Adminisztrátor használati eset (use case) diagram

A bejelentkezést követően az adminisztrátor hozzáférést kap a rendszer szolgáltatásaihoz. Egy admin jogosultságát tekintve rendelkezik mindazzal, amivel egy tag, annyi eltéréssel, hogy egy-egy lekérdezés eredménye több információval szolgál az aktuális

tagról, illetve eszközről. Lehetősége van még módosítani, vagy törölni az adatbázisban szereplő bejegyzéseket, sőt új tagokat és eszközöket is vehet fel. Tagok listázásánál az adminisztrátor az adott tag jelszaván kívül látja annak minden adatát, az eszközök listázásánál pedig ténylegesen mindent. A rendszeradminisztrátor tehát szinte teljes hatalommal rendelkezik a rendszer felett, hiszen a tagok jelszavain kívül mindenhez hozzáfér, módosítási joga pedig mindenre kiterjed.

VI. A rendszer működése

A szoftver – attól függően, hogy az adott felhasználó vendég, tag, vagy adminisztrátor – elkülöníti a jogosultságokat, és ennek megfelelően végezhetnek műveleteket a rendszerben. Folyamatosan számon tartja, hogy ki van belépve a rendszerbe, és minden fontosabb művelet előtt ellenőrzi, hogy az adott felhasználó jogosult-e elvégezni azt. Ebben a fejezetben a szoftver különböző szolgáltatásai kerülnek ismertetésre.

Bejelentkezés

Első lépésként mindenképp be kell lépni, hogy a felhasználó hozzáférjen a rendszer által nyújtott szolgáltatásokhoz. Ez tagok és adminisztrátorok esetén a felhasználói nevük, és jelszavuk megadásával történik. Ha valamelyiket nem adják meg, vagy hibásan írják be, akkor a rendszer egy üzeneten keresztül tájékoztatja őket, hogy az adatok helytelenül lettek megadva, és hogy próbálják meg még egyszer a belépést. Figyelni kell rá, hogy a felhasználói névben és jelszóban szereplő kis- és nagybetűk nem cserélhetők fel.

A bejelentkezés vendégként funkciót bárki használhatja, viszont onnantól kezdve egészen a rendszerből való kilépésig vendégként tekint rá a program, és ennek megfelelően adhat ki utasításokat. Miután a felhasználó bejelentkezett, tájékozódhat róla, hogy a rendszer mely szolgáltatásait veheti igénybe, hogy ezek hol és hogyan érhetők el, valamint betekintést nyerhet az egyes szolgáltatások működésébe is.

Adatok listázása

Tagok és eszközök listázásánál megjelennek az aktuális bejegyzések az adatbázisból sorban egymás alatt, a felhasználó jogosultságának megfelelően. A listázás mindig név szerinti sorrendben történik, viszont lehetőség van csoportosításra: a tagokat státusz szerint, az eszközöket pedig típus, vagy tulajdonos szerint. Ez átláthatóbbá, és a rendszergazdák számára kezelhetőbbé teszi az adatbázist.

Keresés az adatbázisban

A keresési funkció mindig név alapján működik, a találatok pedig ez esetben is abc sorrendben lesznek felsorolva. Csak meg kell adni a nevet, vagy annak egy részét, kiválasztani, hogy tagok, vagy eszközök között keresünk, és az eredmény azonnal látható lesz. A kis- és nagybetűk itt nincsenek megkülönböztetve, ellentétben a bejelentkezéskor is megadott felhasználói névvel és jelszóval. Ha nem adunk meg semmit a tárgyszó mezőben, akkor a kereső minden bejegyzést kilistáz az adatbázisból.

Tag adatainak módosítása

Egy tag adatainak módosítása esetén a rendszergazda a listázás, vagy a keresés eredményéből választhatja ki a módosítani kívánt személyt, majd a változtatások megadását követően a szoftver frissíti az adatokat az adatbázisban. A rendszer segítséget nyújt abban, hogy megmutatja, mely adatok megadása kötelező, vagy ahol nem egyértelmű, megadja a használandó beviteli formátumot (például dátum formátum). A telefonszámnál például nincs különösebb megkötés az adatok bevitelét illetően, hanem azt a rendszer elemzi, és átalakítja a kívánt formátumra. Amennyiben az admin nem töltött ki egy kötelező mezőt, vagy nem megfelelő formátumot adott meg egy helyen, tehát esetleges hiba esetén, a program nem végzi el a módosításokat, hanem figyelmezteti a felhasználót. Ezt követően vissza lehet lépni, javítani a hibás részeket, és újból megpróbálni végrehajtani a műveletet.

Eszköz adatainak módosítása

Egy eszköz adatainak módosítása szinte teljesen megegyezik a tagoknál használt működési elvekkel, viszont itt van egy funkció, amelyre mindenképp ki kell térni. Mivel egy egyesületi eszköz vagy ki van osztva valamelyik tag részére, vagy pedig maga a szervezet a tulajdonos, minden tárgynak lesz konkrét tulajdonosa. Vagy az adatbázisban is szereplő egyik tag, vagy az Egri Vitézlő Oskola. Egy módosítani kívánt tárgy tulajdonosának megadása azonban mégsem kötelező. Ilyen esetekben a rendszer választási lehetőségeket kínál fel. Ez történik akkor is, ha csak a név egy részét írjuk be, és így nem lehetne egyértelműen meghatározni, hogy melyik tagról van szó. Azonos nevű személyek esetén a belépés dátuma segít eldönteni, hogy melyiket válasszuk, így ennek segítségével könnyedén ki tudjuk választani a megfelelő tagot. A választási lehetőségek között megjelenik az Egri Vitézlő Oskola is, mint tulajdonos. Ez a technika működik az eszközök felvétele funkciónál is.

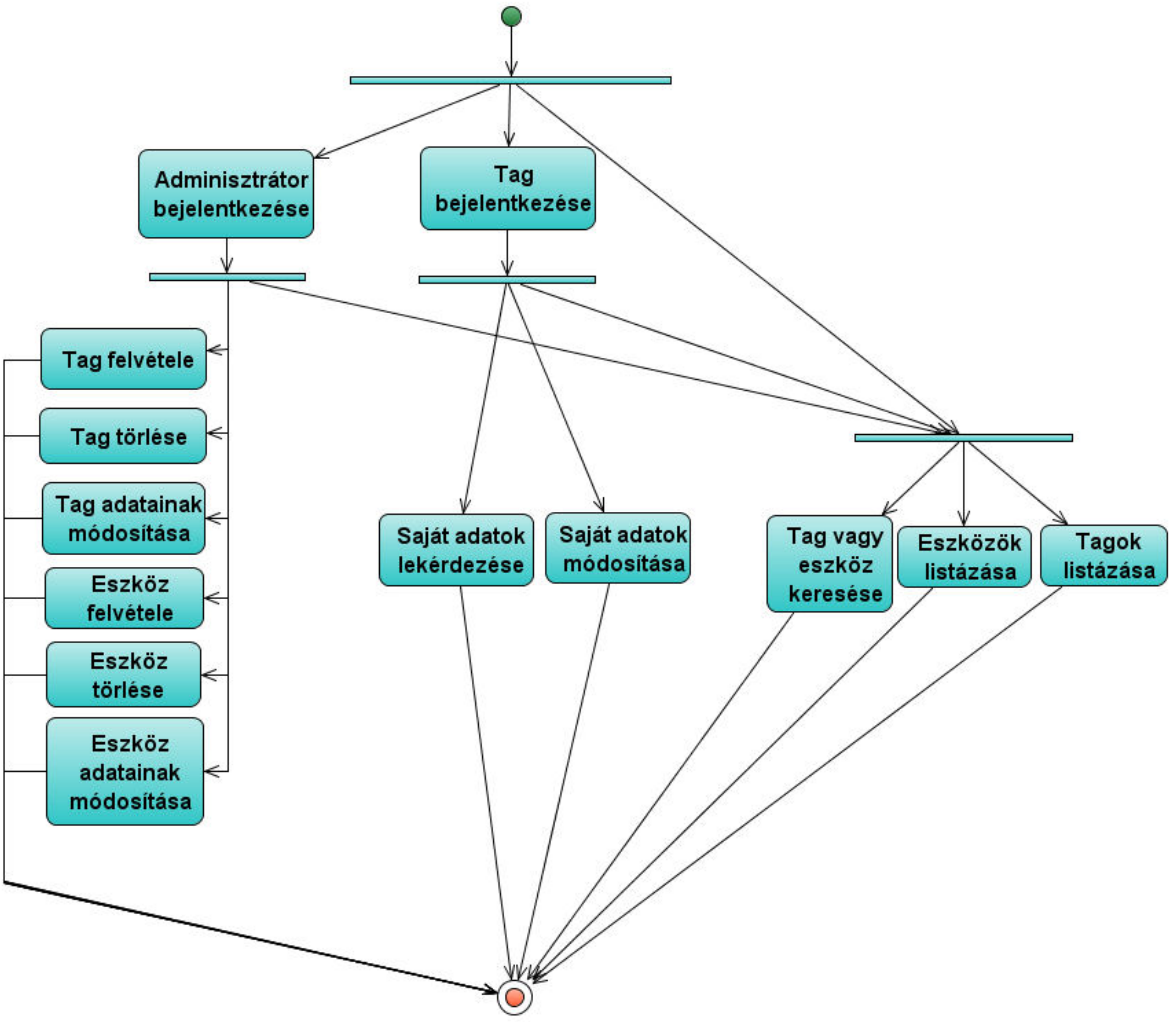
Tag saját adatainak módosítása

Az egyesület tagjai szintén végezhetnek módosításokat a saját adataikon, persze csak korlátozva. Ilyenkor láthatják minden személyes adatukat, és a rendszer jelzi nekik, hogy mi az, amit megváltoztathatnak, és mi az, amihez nem tudnak hozzányúlni. A jogosulatlan módosítást a szoftver nem teszi lehetővé, mivel az ilyen sorokat egyszerűen nem hagyja, hogy megváltoztassák. Az adatok átírása után maga a módosítási művelet ugyanúgy működik, mint az adminisztrátorok esetében a tag adatainak a módosítása.

Jelszó megváltoztatása

Az egyesület tagjainak a rendszerbe való bejelentkezést követően lehetőségük van az addig használt jelszavuk megváltoztatására. Ilyenkor a tagnak meg kell adnia az aktuálisan érvényben lévő jelszavát, majd ezt követően kétszer az új jelszót. Amennyiben nem jól adja meg a régit, nem ír be semmit az újnak, vagy a kétszer megadott új jelszó nem egyezik meg,

akkor a program egy üzenetben közli, hogy mi a probléma. Ilyen esetben a felhasználó visszatérhet az előző oldalra, és ott újra megpróbálhatja végrehajtani a változtatást. Mivel az esetleges visszaéléseket elkerülendően a rendszergazdák nem ismerhetik az egyes tagok jelszavait, viszont megváltoztatni meg kell tudniuk (például ha egy tag elfelejti), az adminisztrátoroknak a rendszer elfogadja, ha nem írnak be semmit a régi jelszó mezőjébe. Viszont ez a funkció csak olyankor működik, ha az admin nem a saját jelszavát akarja megváltoztatni. Tehát ha például egy rendszergazdának kell új jelszót adni úgy, hogy nem tudja régit, akkor abban csak egy másik rendszeradminisztrátor fog tudni neki segíteni. Ez azonban nem probléma, mivel az egyesületben több ember is alkalmas az adminisztrátori pozíció betöltésére, és várhatóan legalább hárman lesznek. Így ha valaki nem tudja a jelszavát, valaki biztosan lesz, aki tud neki segíteni.



4. ábra – Aktivitás- vagy tevékenység (activity) diagram

Törlés az adatbázisból

A rendszergazdáknak lehetőségük van az adatbázisban szereplő bejegyzések törlésére is. Az eszközöket tekintve, nincs probléma ezzel, hiszen egy adott eszköz tulajdonságai között van tárolva, hogy ki az aktuális tulajdonos, így ha töröljük az adatbázisból, akkor a tagtól is egyszerűen „törlődik” az adott eszköz. Tag törlésénél viszont más a helyzet. Egyrészt tekintettel arra, hogy a rendszer folyamatosan fog működni, feltételezhető, hogy mindig lesz benne adminisztrátor, mivel valakinek karban kell tartania az adatbázist. Így egy rendszergazda törlésére csak akkor lehet szükség, ha több adminisztrátor is van, és emiatt egy admin nem törölheti saját magát a rendszerből. Ez tekinthető egyfajta biztonsági intézkedésnek is, hogy esetlegesen még véletlenül se tudja törölni valaki saját magát a rendszerből. A másik fontos kritérium, aminek a program meg kell, hogy feleljen a Vitézlő Oskola elnökségének kifejezett kérése alapján, hogy a tagokat legyen lehetőség csak logikailag törölni az adatbázisból, mivel egy-egy tagnál, az egyesületből való kilépése után is maradhat felszerelés. Ilyen esetekben fizikailag nem töröljük az adatbázisból, hanem csak logikailag, tehát egy változóban (státusz attribútum) jelezzük a megszűnt tagviszonyt. Így továbbra is követhető, hogy mivel tartozik, eszközt visszavételezni pedig ezután ugyanúgy lehet tőle, mint rendes tagok esetén, tehát egyszerűen csak megváltoztatjuk az adott eszköz tulajdonosát. Ilyen esetekben a tag törlésekor a szoftver automatikusan érzékeli, hogy a tagnál még vannak eszközök, és emiatt csak logikai törlést hajt végre, és ezt közli is az adminisztrátorral. A tag státusza innentől kezdve az lesz, hogy „kilépett tag”, és mint ilyet, a tagok listázásánál, vagy keresésénél csak a rendszergazdák láthatják.

Tagok és eszközök felvétele

Az Abydos alapvető funkciója az új tagok és eszközök felvétele, amit természetesen a rendszer adminisztrátorai végezhetnek el. Az eljárás nagyon hasonló a tagok és eszközök adatainak módosításra, ugyanúgy meg lehet adni minden adatot, és szintén vannak megszorítások, hogy mit kötelező. Egy tárgy adatbázisba való felvételekor a tulajdonos megadása ugyanúgy működik, mint az eszköz adatainak módosításakor, tehát egy listából

lehet kiválasztani a megfelelő személyt. Mind tag, mind pedig eszköz adatbázisba rögzítése esetén a szoftver segíti a munkát azzal is, hogy néhány mezőt előre kitölt. Az eszköz- és tagazonosítót a meglévő azonosítók alapján, a beszerzési- és belépési dátumot pedig az aktuális dátumból állapítja meg a rendszer. Ez azért is hasznos egyrészt, mert egy eszköz beszerzése, vagy egy tag belépése után nagy valószínűséggel hamarosan fel lesz véve az adatbázisba, így a dátumhoz nem, vagy csak minimálisan kell hozzányúlni. Másrészt pedig a dátum nem mindig egyértelmű formátuma miatt nem árt, ha ott van egy példa, hogy hogyan kell írni.

A tag adatainak bevitelekor meg kell adni természetesen a leendő felhasználói nevet, valamint jelszót is. A felhasználói nevet illetően vagy a taggal egy előre megbeszélt nevet célszerű megadni, vagy pedig egy tetszőleges nevet, amit aztán a rendszerbe való bejelentkezést követően a tag egyszerűen megváltoztathat. A jelszót itt is kétszer kell beírni, és ha az új tagnak van konkrét kérése, meg lehet adni azt is, viszont hogy az adminisztrátor ne tudja a tagok jelszavait, a rendszer automatikusan generál egyet, amit így az admin sem láthat. Ez egy 6 karakterből (számokból, betűkből vegyesen) álló szó, amit a felhasználói névvel, és némi tájékoztatóval együtt a rendszer a tag felvétele után automatikusan elküld a megadott e-mail címre, vagy ha az nincs megadva, akkor az Egri Vitézlő Oskola, direkt a szoftverhez létrehozott e-mail címére.

Fényképek használata

A nyilvántartásban a tagokhoz és eszközökhöz tartozó adatok mellett követelmény volt, hogy fénykép is szerepeljen a könnyebb azonosíthatóság és jobb informálás érdekében. Mivel az Abydos rendszer nem egy közösségi oldal, ahol a felhasználók kedvükre változtathatják az adatokat, a programot nem kell felkészíteni például a fényképek gyakori megváltoztatásának lehetőségére. Mind a tagok, mind pedig az eszközök tekintetében elmondható, hogy ha egy új személyt, vagy tárgyat felveszünk az adatbázisba, és beállítunk hozzá egy képet, akkor azt utána szükségtelen megváltoztatni. Természetesen képek feltöltését, és törlését biztosítani kell, amire csak az adminisztrátoroknak van jogosultsága, de az esetleges módosítás szintén egyszerűen megoldható. Éppen emiatt a képek nem az adatbázisban, hanem attól függetlenül, a szoftver kódja mellett – az egyéb ikon, és háttérképek között – lesznek eltárolva. Így az

adatbázisból történő lekérdezések gyorsabbak lesznek, hiszen nem kell az esetleg nagyméretű képeket a szervernek az adatbázisból betöltenie. Mindezzel persze egyszerűsödik a program kódja, az adatszerkezet, valamint egyszerűsödik a felhasználói kezelőfelület is. A fényképek kezelése ugyanis nem a minden felhasználó által használt webes elérésű helyről, hanem az egyesület szerveréről lesz karbantartható, amelyhez a rendszergazdáknak korlátlan hozzáférésük van.

Kilépés a rendszerből

Amikor a felhasználó be kívánja fejezni a rendszer használatát, bármikor könnyedén ki tud lépni belőle. A könnyebb kezelhetőség érdekében erre mindig két helyen, az oldal alján és tetején is lehetőség van. Azonban ha csupán bezárja a böngészőt, később akkor sem lehet helyreállítani a kapcsolatot, tehát a program automatikusan kezeli a felhasználók kiléptetését. Ez azért is fontos, hogy illetéktelen személyek így nem juthatnak hozzá bizalmas információkhoz. A honlapról való elnavigálás után ezért tehát újra be kell lépni a főoldalról ahhoz, hogy valaki ismét igénybe vegye a rendszer szolgáltatásait.

„Vissza” funkció

Nehéz meghúzni a határt a között, hogy a szoftver kezelőfelületét megfelelően lehessen használni, tehát hogy felhasználóbarát legyen az oldal, valamint hogy megfelelő funkcionalitás is legyen beleépítve a programba. Ez esetben a funkcionalitást háttérbe helyezve a rendszer nem tartalmaz „vissza” gombokat. Ezek olyankor lennének hasznosak, amikor például egy módosítás, vagy tagfelvétel nem sikerül, akkor ne keljen előröl kezdeni az adatok megadását. Ezt a funkciót azonban a böngészőkbe alapból beépített „vissza” gombbal el lehet érni, méghozzá 100 százalékig megfelelő minőségben.

VII. Az adatbázis

A létrehozni kívánt rendszer elsősorban adatok tárolását hivatott szolgálni, és hogy azokat a felhasználó könnyedén kezelni tudja. Nem kell ismerni az alkalmazott adatbázis leíró nyelvet, vagy nem kell informatikusnak lennie valakinek, hogy megfelelően tudja használni az alkalmazást. A program kezelőfelületén keresztüli használat mindenki számára könnyen elsajátítható.

Az adatbázis azonos jellemzőkkel rendelkező strukturált adatok összessége, amelyet egy tárolásra, lekérdezésre és szerkesztésre alkalmas szoftver kezel. Az adatbázisok célja adatok megbízható, hosszú távon tartós (idegen szóval perzisztens) tárolása, és viszonylag gyors visszakereshetőségének biztosítása. Adatbázis kezelő rendszernek (DataBase Management System) azt a szoftvert nevezzük, amely az adatok felhasználását lehetővé teszi. Ez egy több felhasználós, hálózatos környezetben működő, adatbázisokhoz való hozzáférést, és a felhasználói folyamatok zavartalan működést biztosító szoftveralkalmazás. Funkciója továbbá egy adatbázis szerver biztosítása, amin maga az adatbázis lesz elérhető a felhasználók számára, amely egy úgynevezett kliens-szerver modell alapon működik.

Az Abydos mögött egy adatbázis áll, amely az egyesület életében megjelenő tárgyakat, és embereket képes nyilvántartani. A valós világban megjelenő konkrét tagokat és eszközöket, az adatbázis tartalmának feleltethetjük meg. Ahhoz, hogy a felhasználók igénybe tudják venni a szoftver nyújtotta szolgáltatásokat, először az adatbázist kell elkészíteni, azonban ezt is több részre lehet bontani.

Az adatbázis létrehozása a következő lépésekből áll:

1. Elemzés, tervezés, modellezés, melynek végén előáll a séma
2. A megfelelő eszközrendszerrel leírni a sémát, így megadva az üres fizikai adatbázis szerkezetét.
3. A fizikai adatbázis feltöltése a szerkezetnek megfelelően. A karbantartás szintén az itt használt eszközrendszerrel történik.
4. Adatok lekérdezése az adatbázisból.

Az adatbázis létrehozásához szükséges első lépés a valós rendszer, azaz az Egri Vitézlő Oskola megfigyelése, megismerése. Erre azért kell nagy hangsúlyt fektetni, mert véleményem szerint ahhoz, hogy valaki egy jól működő, az életben is helytálló versenyképes rendszert tudjon létrehozni, ismernie kell a rendszer egészének működését. Tekintve, hogy én is tagja vagyok az egyesületnek, első kézből tudom, hogy mit kell tudnia az általam készített szoftvernek.

Az első fázis az elemzés, ahol egyrészt fel kell térképezni, hogy mi mindent kell majd nyilvántartani a tagokat, illetve eszközöket illetően, másrészt pedig hogy milyen módszerrel. Az egyesület tagjairól nem csupán a hétköznapi adatokat kell eltárolni, mint például az elérhetőségek, hanem részletes személyes adatokat is. Az Oskola elnökének elmondása alapján a kifejlesztendő rendszer létrehozásának célja nem csak az informálás. Egyes külföldi utak esetén például pontos utas listát kell készíteni, amelyen fel kell tüntetni a tagok több személyes adatát is. Vagy egy biztosítás megkötéséhez, tudni kell például az emberek TAJ számát, ezért a rendszernek ezen információk kinyerésére is alkalmasnak kell lennie.

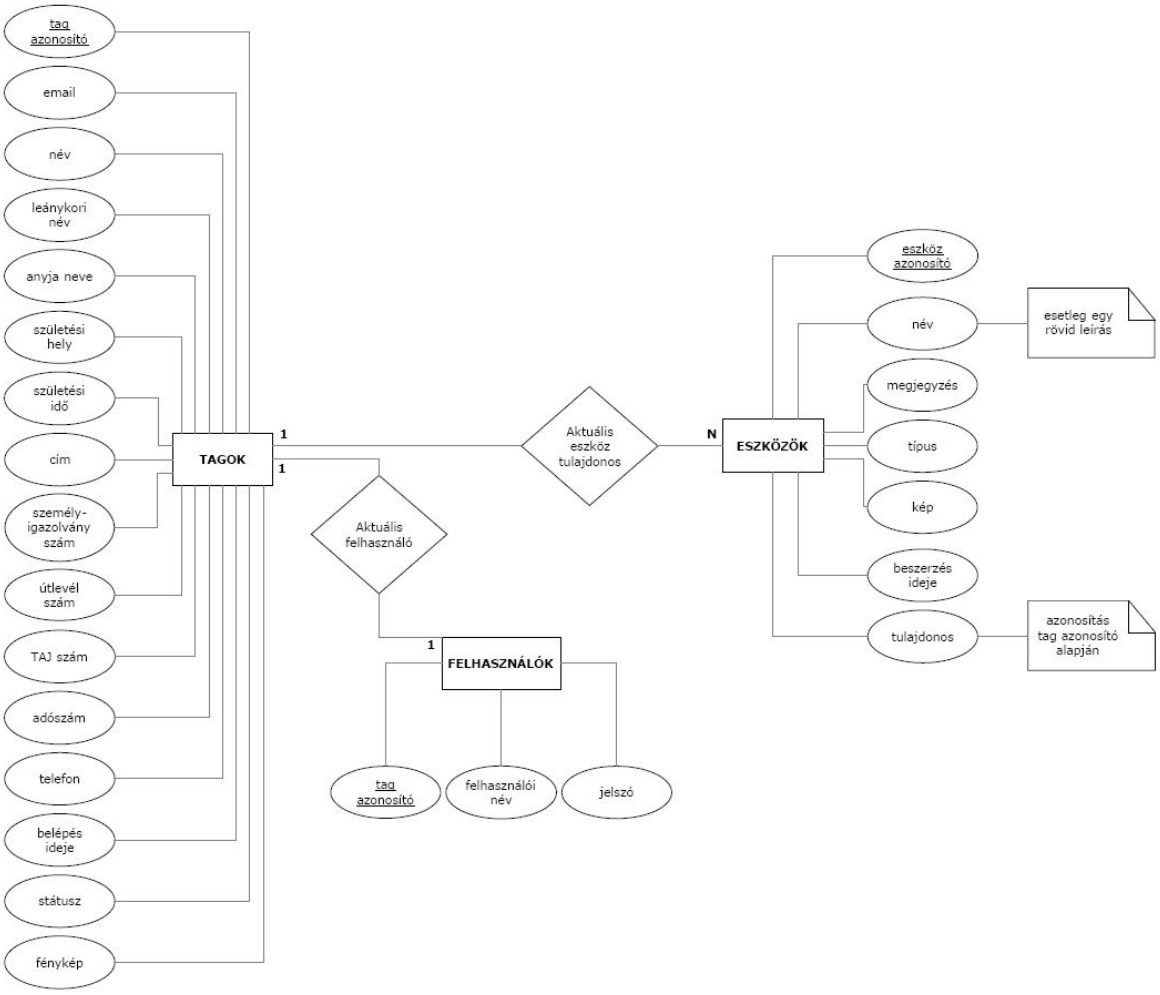
Az adatbázis felépítését egy úgynevezett sématervező eszközzel, az ER-modell (Entity-Relationship model – Egyed-kapcsolat modell) segítségével írom le, amihez azonban előbb ismertetek néhány hozzá kapcsolódó definíciót.

Egyed, entitás (entity) ami azonosítható, és megkülönböztethető egymástól, a rendszer alapeleme. Ide sorolható egy-egy tag, vagy eszköz. A jele egy téglalap.

Tulajdonság (attribútum) az egyedek jellemzői, például név, vagy státusz. A jele egy ellipszis.

Kapcsolat (relationship) az egyedek közti viszonyt írja le, például hogy egy eszköznek melyik tag a tulajdonosa. A jele egy rombusz.

Az ER-modell egy adatbázis modellező eszköz, segítségével az adatbázist átláthatóbban lehet leírni, és az adatok struktúrájába nyerhetünk ezen keresztül bepillantást. Ily módon az adatbázisban szereplő táblák létrehozása leegyszerűsödik, hiszen átláthatóbbá válik, hogy mit kell nyilvántartani, vagy hogy az adatok hogyan kapcsolódnak egymáshoz. Mindez abban is segít, hogy az éppen aktuális adminisztrátorok jobban megértsék az adatok struktúráját, és magát a rendszert.



5. ábra – ER-modell

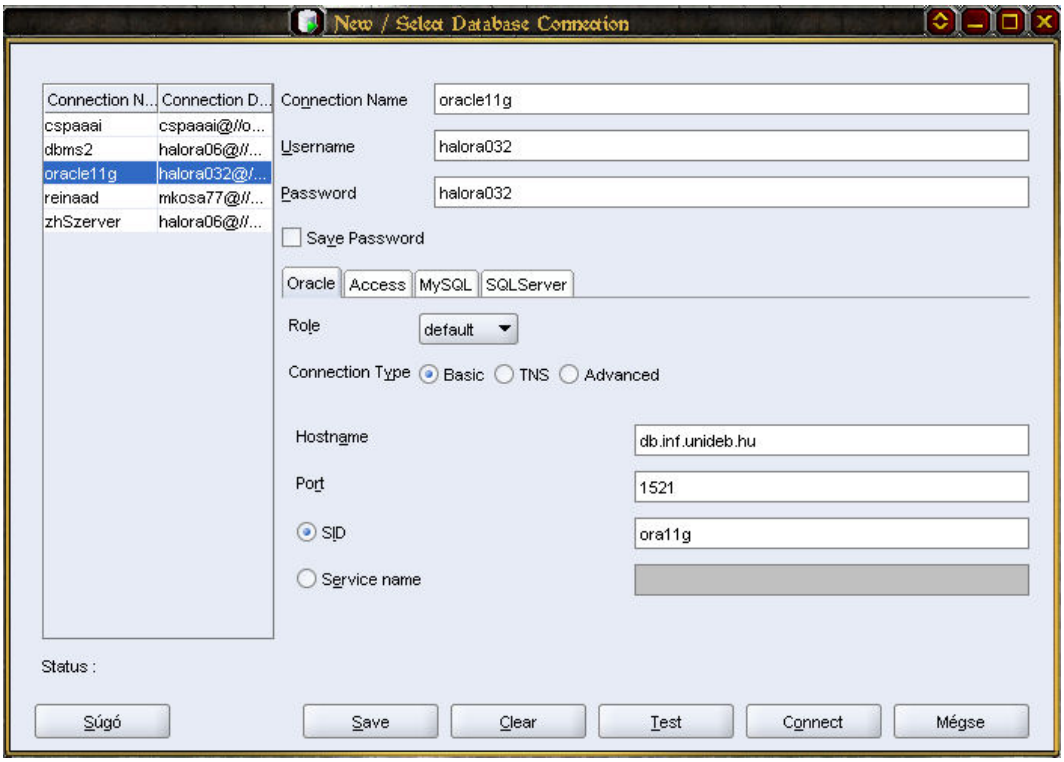
Az adatbázis konkrét létrehozásához célszerű volt egy olyan adatbázis szervert alkalmazni, amit össze lehet egyeztetni a Java technológiával. Mivel a szoftveremet a NetBeans 6.5-ös verziójával fejlesztettem, és abban van beépített adatbázis kezelő rendszer, úgy döntöttem, hogy azt fogom használni. A Java DB egy nyílt forráskódú, 100 százalékig Java technológián alapuló Apache Derby relációs adatbázis kezelő rendszer. Ez amellett, hogy teljes mértékben megfelel az elvárásoknak, könnyű összeegyeztetni az ugyancsak Java nyelven írt alkalmazással, és ráadásul integrálva van a Netbeans-be. Amennyiben ez valakinek mégsem felelne meg, és külső adatbázis szervert szeretne használni, úgy a program arra is lehetőséget biztosít. Ezt egy interfész segítségével valósítja meg, amelyben ki lehet cserélni, hogy a program melyik szervert használja. Alaphelyzetben az Apache Derby-n kívül a Debreceni Egyetem által is használt Oracle szerver van benne másodlagos kiszolgálóként, azonban ezt még tovább lehet bővíteni az igényeknek megfelelően.

```
public class connect {
    public static String jdbc = derby.jdbc;
    public static String usr = derby.usr;
    public static String pwd = derby.pwd;
    public static Connection connect() throws SQLException {
        Connection con = DriverManager.getConnection(jdbc, usr, pwd);
        return con;
    }
/*  public static String jdbc = oracleUnideb.jdbc;
    public static String usr = oracleUnideb.usr;
    public static String pwd = oracleUnideb.pwd;
    public static Connection connect() throws SQLException {
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Connection con = DriverManager.getConnection(jdbc, usr, pwd);
        return con;
    }*/
}
```

Az Abydos üzembe helyezése előtt első lépésként magát az adatbázist kell létrehozni. Ehhez a Netbeans egy könnyen kezelhető grafikus felületet biztosít, ahol meg kell adni az adatbázis nevét, valamint a kapcsolódáshoz szükséges felhasználói nevet és jelszót. Ezek kötöttek, hiszen a programba alapból bele van írva, így ezt nem kell később manuálisan megadni.

6. ábra – Apache Derby adatbázis létrehozás

Oracle szerver használata esetén az eljárás hasonlóan működik. Az adatbázis létrehozását követően kapcsolódni kell hozzá, hogy létrehozhassuk a táblákat. Az Abydos használata során a kapcsolat kiépítését és bontását a program végzi, így akkor már nekünk ezzel nem kell törődni. Erre most azért van szükség, mivel az adatbázis még nem tartalmaz egyetlen embert sem, nincs még adminisztrátor, aki felvehetné a többi tagot. A Netbeans-ben az adatbázisok között már ott szerepel az abydos is, amelyhez könnyedén tudunk csatlakozni, majd az utasítások végrehajtása funkciót használva hozhatjuk létre, és tölthetjük fel adatokkal a táblákat. Oracle szervert használva szinte ugyanez a szisztéma, habár ott az adatbázishoz való kapcsolódáshoz szükség van például az SQL Developer nevű programra. Létre kell hozni egy új kapcsolatot, ahol meg kell adni a felhasználói név, és jelszó mellett egy URL-t, egy port számot, valamint egy SID-et (System ID).



7. ábra – Kapcsolódás Oracle adatbázishoz

Ezt az egész procedúrát bele lehetne építeni a program kódjába is, azonban mivel ezt csak egyszer – a szoftver üzembe helyezésekor – kell elvégezni, és nem is jár túl nagy munkával, úgy gondolom, hogy nem szükséges.

A táblák létrehozásához csupán az előre meghatározott SQL utasításokat kell lefuttatni, aminek eredménye képen létre is jönnek a megfelelő struktúrával rendelkező táblák. Arra mindenképp oda kell figyelni, hogy a kétféle adatbázis szerver más-más szintaktikát használhatnak bizonyos esetekben (így például a dátum megadásánál), ezért a két szerverhez két külön utasítássorozatot (szkriptet) készítettem. Az adatbázis ezzel tehát készen van, el lehet kezdeni az adatok feltöltését. Erre is van egy minta, melynek segítségével – az SQL nyelv ismerete nélkül is akár – el lehet végezni a műveletet. Minimum egy rendszergazda jogosultságú tagot fel kell venni, hogy ő aztán el tudja végezni a további tagok és eszközök felvételét. Ezt úgy lehet, hogy a tag státusza után vesszővel elválasztva oda kell írni, hogy „admin". Az utasítás sikeres elvégzése után már be is lehet jelentkezni a rendszerbe, és onnantól a teljes funkcionalitás használható. Ha úgy kényelmesebb, az imént kiadott utasítás átírásával is fel lehet venni tárgyakat, és személyeket az adatbázisba.

Tag felvétele Apache Derby adatbázis szerver használata esetén:

```
insert into Tagok values('Csillag Péter','elnökségi tag, admin',
    '2000-02-10',
    'cspsg1@yahoo.co.uk','3300,Eger,Széchenyi út 16.','205034929','',
    'Szabó Erzsébet','Eger','1984-03-03',
    'szigsz_1','utlevel1','TAJszam1','adoszam1',1);
insert into Felhasznalok values('user1','alpha',1);
```

Tag felvétele Oracle adatbázis szerver használata esetén:

```
insert into Tagok values('Csillag Péter','elnökségi tag, admin',
    TO_DATE('2000-02-10','YYYY-MM-DD'),
    'cspsg1@yahoo.co.uk','3300,Eger,Széchenyi út 16.', '205034929','',
    'Szabó Erzsébet','Eger',TO_DATE('1984-03-03','YYYY-MM-DD'),
    'szigsz_1','utlevel1','TAJszam1','adoszam1',1);
insert into Felhasznalok values('user1','alpha',1);
```

Eszköz felvétele Apache Derby adatbázis szerver használata esetén:

```
insert into Eszkozok values('Sötét bordó dolmány','ruha',
    '2005-06-13',1,'Felújított.',1);
```

Eszköz felvétele Oracle adatbázis szerver használata esetén:

```
insert into Eszkozok values('Sötét bordó dolmány','ruha',
    TO_DATE('2005-06-13','YYYY-MM-DD'),1,'Felújított.',1);
```

Az ER-modell alapján is látható, hogy az adatbázisban 3 táblát fogok használni az adatok tárolására. Egy tábla a tagok számára, egy az eszközöknek, egy külön tábla pedig a tagok felhasználói neveinek és jelszavainak a tárolására. Ezt azért kell így megoldani, mert a követelmények között szerepel, hogy a regisztrált felhasználók jelszavait külön, bizalmas információként kell tárolni, így ez talán valamivel nagyobb biztonságot ad. A táblák mezőinek – amennyire lehetett – beszédes neveket adtam, hogy az adatbázist látva, vagy a program szövegét olvasva az ember könnyebben megértse, miről is van szó. A következőkben ismertetem az általam használt táblákat, és azok tulajdonságait.

A táblákban a következő adattípusokat használom:

VARCHAR(x) Karakteres típus, ahol az x egy egész szám, ami megadja, hogy maximum hány karakter hosszú lehet a szó.

NUMERIC Numerikus (szám) típus.

DATE Dátum típus

Tagok tábla

A tábla az Egri Vitézlő Oskola tagjainak adatainak tárolását hivatott szolgálni. Eredetileg az elsődleges kulcsot illetően – ez egy olyan attribútum, ami minden személyt egyértelműen meghatároz – az e-mail címet gondoltam használni, mivel abból nem lehet két egyforma. Azonban végül két ok miatt is, bevezettem egy külön, numerikus típusú azonosítót mind a tagoknál, mind pedig az eszközöknél. Az egyik ok, hogy a tagságot tekintve nem mindenki rendelkezik e-mail címmel, így az nem lehetne egyedi. Másrészt pedig egy adatbázisban törekedni kellene a minél magasabb normálforma elérésére, azonban a tranzitív (belső) függések miatt ez már nem megoldható. Ez olyankor alakul ki, amikor egy nem

elsődleges kulcs attribútum egyértelműen meghatározza az egyedeket. Több olyan egyedi adatot is tárolni kell, ami miatt fennáll ilyen jellegű függés (például a személyi igazolvány szám), emiatt a tábla 2. normálformában van.

```
create table Tagok (
  nev varchar(30) not null,
  statusz varchar(20) not null,
  belepesDatuma date not null,
  email varchar(30),
  cim varchar(50),
  telefon varchar(9),
  leanykoriNev varchar(30),
  anyjaNeve varchar(30),
  szuletesiHely varchar(20),
  szuletesiIdo date not null,
  szIgSzam varchar(8),
  utlevelSzam varchar(8),
  tajSzam varchar(8),
  adoSzam varchar(8),
  tagId numeric primary key
);
```

Felhasználók tábla

A felhasználók tábla nevéből is kitűnik, hogy az egyesület tagjairól tárol bizonyos információkat. Az itt lévő adatokra különös biztonsággal kell figyelni, hiszen rendkívül fontos, hogy ne kerülhessenek illetéktelen kezekbe. Az itt szereplő elsődleges kulcs megfeleltethető a tagok táblában lévővel, mivel mindkét helyen ugyanarra a személyre vonatkozik. Éppen ezért, ha az egyik helyen megváltozik egy tagazonosító, akkor a másik táblában is módosítani kell, azonban ezt a rendszer figyeli. A felhasználói nevek egyedisége miatt itt is fennáll a belső függés, emiatt ez a tábla is 2. normálformában van.

```
create table Felhasznalok (
  felhNev varchar(20) unique,
  jelszo varchar(20) not null,
  tagId numeric primary key
);
```

Eszközök tábla

Az adatbázisban lévő eszközök ebben a táblában vannak eltárolva, ugyancsak numerikus típusú elsődleges kulccsal. A tárgyakat tehát ugyanúgy számok alapján lehet azonosítani, mint a Vitézlő Oskola tagjait, de ezt nem szabad összekeverni. A szoftverben megfelelően el van különítve a kettő, ezért nem szükséges számok helyett például t, vagy e betűvel kezdődő szöveges azonosítókat használni.

Az eszközöknél szerepel egy tulajdonos mező is, amely egyfajta hivatkozásként funkcionál, és egy tag tagazonosítójával azonos értéket tartalmaz. Ennek segítségével egyértelműen meg lehet határozni, hogy ki az adott eszköz tulajdonosa. Mind a tagok, mind pedig az eszközök esetén az azonosítók számozása egytől kezdődik, és lényegében bármilyen pozitív egész számot értékül lehet nekik adni. Mint arról már az előző fejezetben is volt szó, egy-egy eszköznek nem feltétlen van konkrét tulajdonosa, ezért ilyenkor abban a mezőben egy 0 érték szerepel, amihez a program automatikusan az Egri Vitézlő Oskolát rendeli. A táblában szereplő egyetlen egyedi érték maga az azonosító, így emiatt a tábla 3. normálformában van.

```
create table Eszkozok (
  nev varchar(120) not null,
  tipus varchar(10) not null,
  beszerezve date not null,
  tulajdonos numeric,
  megjegyzes varchar(120),
  eszkozId numeric primary key
);
```

Összefoglalva tehát az adatbázis 2. normálformában van, további normalizálás pedig nem szükséges, mert úgy gondolom, hogy ettől függetlenül megfelelő adatbázist sikerült létrehoznom. Mindenek előtt a legfontosabb dolog, hogy a követelményeknek eleget tesz, továbbá pedig logikus felépítésű, és könnyen használható. A rendszer üzembe helyezését követően, a felmerülő igények szerint tovább fejleszthető, vagy akár le is cserélhető a teljes adatbázis szerver.

Megszorítások

Az adatbázisban szereplő táblák létrehozásakor főként kétféle megszorítást kellett alkalmazni. Táblánként egy-egy attribútum, mint elsődleges kulcs szerepel, amihez a „PRIMARY KEY" megszorítás tartozik. Ez egyedivé teszi, és nem engedi, hogy null értéket vegyen fel az adott attribútum. A másik, több helyen használt megszorítás a „NOT NULL", melynek használatánál mindig kell értéket adni, vagyis a mezőt nem lehet üresen hagyni. Továbbá a felhasználók táblában lévő felhNev attribútum – amelyben a tagok felhasználói nevét tároljuk – „UNIQUE" megszorítással rendelkezik. Erre azért van szükség, mert a követelmények szerint a rendszernek figyelnie kell, hogy a tagok felhasználói nevei egyediek legyenek, hogy ne fordulhasson elő két egyforma.

A táblák közti kapcsolat alapját a mindegyikben szereplő tagazonosító adja, célszerű lenne tehát azt külső kulcsként alkalmazni a felhasználók és az eszközök táblában. Azonban több érv, és ellenérv megvizsgálása után úgy döntöttem, hogy a táblák ne tartalmazzanak külső kulcsokat, hanem majd a program fogja kezelni a hivatkozási integritási megszorításokat. Mivel a tagazonosítót is lehet módosítani, külső kulcsok használata esetén az adatbázis tábláiba kellene tenni olyan megszorítást, hogy a Tagok táblában lévő tagId megváltoztatásakor írja át a másik két helyen is a megfelelő attribútum értékeket (CASCADE). Ez azért nem megfelelő, mert a rendszer több adatbázis szerverrel is képes együttműködni, és így nem lehet garantálni, hogy ez a funkció mindig megfelelően fog működni. Viszont ha ezt maga a program csinálja, akkor – ha az jól meg van írva – biztosan megfelelő lesz a működés minden esetben. Az Abydos-ban tehát különösen nagy figyelmet kap a tagazonosító módosítása, mivel ez az egyetlen művelet, amely sértheti a megszorítást. Így annak megváltoztatása esetén a rendszer közvetlen egymás után hajtja végre a

frissítéseket mindhárom táblában, és ha az egyik nem sikerülne valami miatt, akkor visszagörgeti az egész tranzakciót (ROLLBACK), így a módosítás egyszerűen nem hajtódik végre. Valójában minden az adatbázisra vonatkozó módosítási művelet így működik, vagyis egy módosításra irányuló tranzakció-sorozat előtt az automatikus végrehajtás kikapcsol (*con.setAutoCommit(false)*), majd miután a program megpróbálta elvégezni a változtatásokat, visszakapcsol (*con.setAutoCommit(true)*). A műveletek tehát teljesítik a következőkben felsorolt tranzakciókra vonatkozó kritériumokat.

Tranzakció (transaction) az adatbázis-műveletek végrehajtási egysége, amely DML utasításokból áll, és a következő tulajdonságokkal rendelkezik:

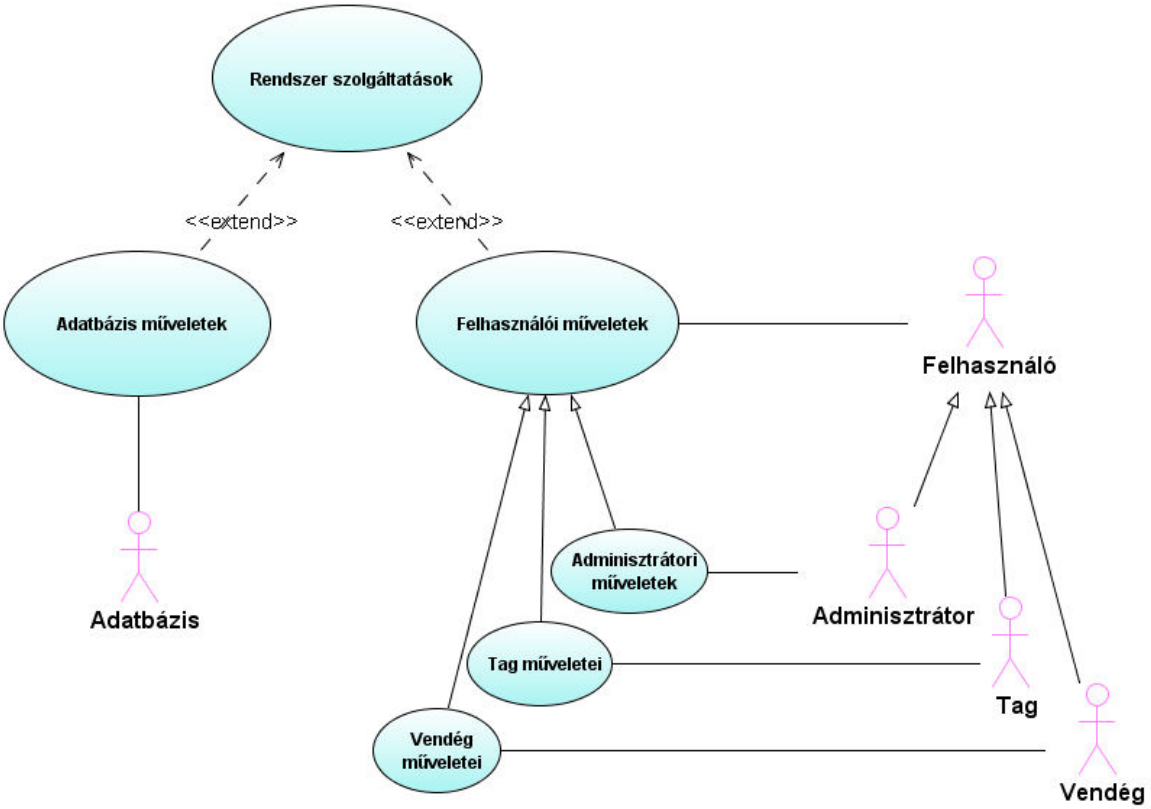
- *Atomitás* (**a**tomicity) a tranzakció legyen egy és oszthatatlan (vagy teljesen végrehajtjuk, vagy egyáltalán nem).
- *Konzisztencia* (**c**onsistency) őrizze meg az adatbázis konzisztenciáját, azaz a tranzakció sikeres lefutása esetén az adatoknak konzisztens állapotból konzisztens állapotba kell kerülniük.
- *Izoláció* (**i**solation) minden tranzakciónak látszólag úgy kell lefutnia, mintha ez idő alatt semmilyen másik tranzakciót nem hajtanánk végre.
- *Tartósság* (**d**urability) egy sikeres tranzakció lefutásának eredményének tartósnak kell lennie (hiba esetén sem változik az eredmény).

VIII. Technológia és megvalósítás

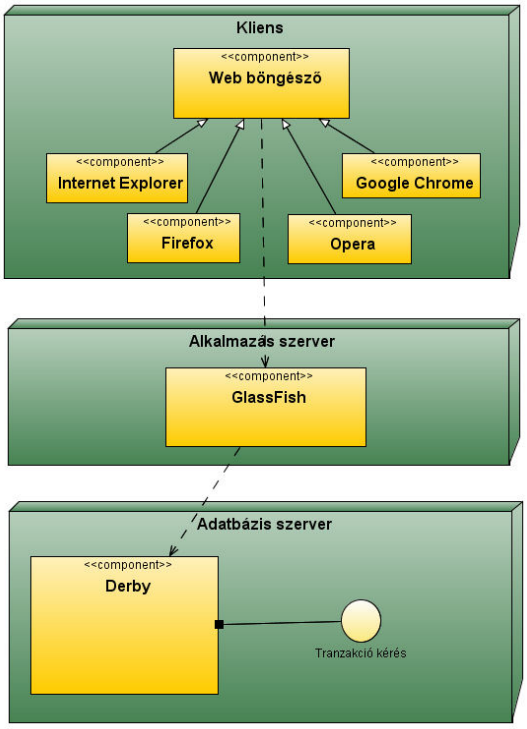
Minden valamire való szoftver bonyolult lépések sorozatán keresztül jut el arra a pontra, hogy aztán üzembe lehessen állítani, megkezdve a használatát. Egy jól működő szoftver alapja a megfelelő követelmény feltárás és elemzés, ami rendkívül fontos, hiszen nem kevés időt, és adott esetben pénzt spórolhatunk meg vele. Ebbe a folyamatba a megrendelőt is célszerű bevonni, aki meg tudja mondani pontosan, hogy majd mit vár el a működőképes rendszertől, vagyis hogy miért fizet. A követelményeket dokumentálni kell, így megelőzhetjük a félreértéseket, és számtalan reklamációt. A megrendelő, és a kivitelező is jobban jár, ha a szoftver az elvárásoknak megfelelő. Nagy hangsúlyt fektettem a követelmények feltárására, amelyhez nagy segítséget nyújtott az UML (Unified Modeling Language – Egységes Modellező Nyelv). Az UML, az objektumorientált programozás szabványos specifikációs nyelve, grafikus jelöléseket használ a rendszerek absztrakt modelljének leírására. Az UML diagrammokat Netbeans 6.5-ben készítettem el, „drag and drop” módszer segítségével.

A Use Case, vagy más néven használati eset diagram segítségével a rendszer funkcionalitását lehet modellezni, könnyen értelmezhető, viszonylag egyszerű ábrák használatával. A követelmények leírására és pontosítására szolgáló grafikus modellező eszköz, amit a szoftverfejlesztés elején használunk. A diagram leírja a rendszert, és az azt körülvevő külső szereplők (például a felhasználók, és az adatbázis) közti kapcsolatot. A középpontban a rendszer által végrehajtott funkciók vannak. Use Case diagramokat használtam például az V. fejezetben a forgatókönyvnél, ahol jogosultság szerint bemutattam, hogy melyik felhasználó milyen szolgáltatásokat vehet igénybe.

Az aktivitás, vagy tevékenység (angol terminológiával Activity) diagram (lásd VI. fejezet – 4. ábra) a rendszer dinamikus viselkedését írja le az egymást követő eseményeket bemutatva, szemléltetve ezzel az időbeli sorrendet. Minden, a rendszer működése közbeni folyamatot modellezni tudunk ennek segítségével, azonban nem célszerű túl részletesre készíteni. Nem kell minden egyes folyamatot szétbontani külön részekre, hiszen a túlzott szemcsézettség csak megbonyolítja a diagram érthetőségét és átláthatóságát. Általában a főbb szolgáltatásokat szokás lemodellezni, az alkalmazás használatának kezdetétől, annak befejezéséig.



8. ábra – Birdview (a rendszer madártávlatból)



9. ábra – Deployment (alkalmazás) diagram

A Deployment, vagy más néven alkalmazás diagram használatával a program implementációja modellezhető le. A felhasznált hardvert, a hardverre telepített szoftverkomponenseket és azok viszonyát lehet reprezentálni ennek a technikának a felhasználásával. A diagram csomópontokat (Nodes) definiál, amelyek vagy egy konkrét hardver eszközt, vagy valamilyen szoftver környezetet reprezentálnak. A mai alkalmazások tekintetében elmondható, hogy a sokféle architektúra, és rengeteg különböző modell miatt az alkalmazás diagram által nyújtott specifikáció nagyban megkönnyíti a rendszerfejlesztést.

Ahogy az alkalmazás diagramból is látszik, az Abydos egy 3 rétegű webes alkalmazás, amelyhez szükségem volt az adatbázis szerveren kívül egy alkalmazás szerverre is. Ez egy olyan kiszolgáló program, amely különféle alkalmazásokat oszt meg számos kliens között. Átveszi az erőforrás igényes alkalmazásokat a kliens oldalról, és áthelyezi egy köztes szintre, az alkalmazás szerverre. Erre a célra a NetBeans-be alapból beépített GlassFish V2-t, a nyílt forrású vállalati alkalmazás szervert használtam.

A rendszer fejlesztése

A rendszer fejlesztése során az evolúciós, vagy prototípus rendszerfejlesztési modellt használtam, melyen belül is a feltáró prototípus készítését választottam. A modell segítségével a fejlesztés fázisait egymással párhuzamosan lehet végrehajtani, így a követelmények teljes, és részletes kidolgozása nélkül el lehet kezdeni a konkrét fejlesztést. A feltáró prototípus készítés lényege, hogy ha megértettük a követelmények egy részét, akkor egy annak megfelelő alkalmazást elkészítünk, így előáll egy korlátozott funkcionalitású prototípus. A követelmények folyamatos alakításával finomítjuk magát a szoftvert is. A prototípus segítséget nyújt abban, hogy a megrendelő, és a leendő felhasználók még a rendszer teljes elkészülte előtt megtekinthetik, hogy hogyan is működik, így jobban előjönnek az esetleges hibák, és könnyebbé válik a fejlesztés. Ennek segítségével nagyon hamar egy működő szoftvert kapunk eredményül. Részemről én próbáltam az alapszolgáltatásokkal kezdeni a fejlesztést, amit mindenki használhat, és így haladtam a bonyolultabb funkcionalitás felé. Végleges formába akkor kerül az alkalmazás, amikor már minden követelménynek megfelel, és nincsenek újabb elvárások. A szoftver üzembe állítását követően felmerülő újabb elvárások később is integrálhatók a rendszerbe.

A követelmények feltárásakor többféle módszert is igénybe vettem a minél jobb eredmény érdekében. Az egyik az interjú készítése. Igazság szerint ezt nevezhetnénk egyfajta közvélemény kutatásnak is inkább, mivel a rendszer leendő felhasználóit, az úgynevezett kulcsfigurákat kérdeztem meg a szoftver működésével kapcsolatos igényeikről. A kulcsfigurák nézőpontjai gyakran eltérnek, sőt esetleg szemben is állhatnak egymással. Ezeket megfelelően össze kell egyeztetni, ezért megpróbáltam minél több embert megkérdezni a témával kapcsolatban, hogy a több vélemény alapján könnyebb legyen döntést hozni. Az interjúk nyíltak voltak, tehát egyszerű beszélgetéseket folytattam le a végfelhasználókkal. Végül az eredményeket a megrendelővel közösen elemeztük, és meghatároztuk a végső követelményrendszert. Interjú készítésénél probléma lehet, hogy az alanyok, a számukra triviálisnak tűnő dolgokat nem említik meg, és ha a fejlesztő ezekkel nincs tisztában, akkor a szoftver nem lesz az elvárásoknak megfelelő. Jelen esetben azonban ez nem jelentett gondot, hiszen ismerem a valós rendszert, mivel magam is részt veszek benne.

A másik, általam használt technika a követelmények feltárására az etnográfia, azaz a megfigyelés. Ennek segítségével betekintést nyerhetünk, hogy a kulcsfigurák hogy használják az alkalmazást, így következtetéseket vonhatunk le, hogy hogyan fejlesszük tovább a rendszert. Ennél a technológiánál nagy segítséget nyújt a prototípus alapú fejlesztés, mivel egy korlátozott funkcionalitású programot már ki tudnak próbálni a felhasználók. Megfigyeléskor vizsgálni kell, hogy a funkciók mennyire megfelelőek, vagyis hogy van-e olyan, amit nem használnak, vagy olyan, ami hiányzik. Fontos, hogy mennyire egyértelműek a szolgáltatások, tisztában vannak-e a használatukkal, és figyelni kell, hogy például nem bosszankodnak-e valamin. A megfigyelt információkat ezt követően elemezni kell, és a megrendelővel közösen felülvizsgálni a követelményeket. Ez a fázis tekinthető egyfajta tesztelésnek, ahol a felhasználók validációs-, és hiányossági tesztelést is végrehajtanak. Az előbbivel megpróbáljuk belátni, hogy a program megfelel a felhasználó elvárásainak, az utóbbi módszerrel pedig megpróbálunk fényt deríteni az esetleges hibákra. Tesztelésre természetesen a szoftver végleges formájába kerülése után is szükség van, amit a fejlesztő informatikus fog elvégezni, hogy megbizonyosodjon róla, hogy a rendszer mindennek eleget tesz, aminek kell.

A rendszer architektúrája

Egy jól működő szoftverhez egy jól megtervezett architektúrára van szükség. Ha ezt jól megcsináljuk, akkor valószínűleg megfelelő lesz a működése. Architektúrális tervezésnél általában stílusokat, mintákat használunk fel, így nagyon erős az architektúra elemek újrafelhasználhatósága. Több, ilyenkor felhasználható minta is szóba jött. Végül az Abydos rendszer fejlesztésénél a rétegzett modellt részesítettem előnyben, hiszen a szoftver rendeltetéséből, és működéséből kifolyólag ez a legcélszerűbb. Ebben a modellben egy olyan architektúra van, ahol rétegek képzelhetők el egymás alatt, és egy réteg csak az alatta, vagy felette levő réteggel van kapcsolatban, tehát csak azzal kommunikál. Ez a technika ma már elég általános, mivel nagyon sok rendszer fejlesztenek ennek a segítségével. A modell rendkívül jól összeegyeztethető az evolúciós prototípus alapú fejlesztéssel, amit én is alkalmazok. Ezen kívül még számos előnye van ennek a technikának, hiszen például teljesítménynövelő hatása van, mert a középső réteg tehermentesíti a klienseket, valamint a biztonság is nagy szerephez jut az ilyen alkalmazásokban. Másik nagy előnye pedig, hogy megkönnyíti a hordozhatóságot, és az aktuális, valamint a jövőbeli fejlesztéseket, mivel a rétegek valamennyire elkülöníthetők egymástól. A rendszer kevésbé nagy mérete miatt a modell megfelelően használható.

Az Abydos tehát egy 3 rétegű webalkalmazás, amelyben – mint más, ezt a technológiát alkalmazó rendszerben – megtalálható az első szinten az adatbázis, második szinten az üzleti logika, a harmadik szinten pedig a megjelenítési réteg.

Kliens (böngésző) ←→ Alkalmazás szerver ←→ Adatbázis szerver

A legalsó réteg kezeli az adattárolást, az alkalmazás szerver által kért adatokat szolgáltatja a kapcsolódó adatbázisból. Az adatok valamilyen nyers formában vannak tárolva, beolvasás után átadja a középső rétegnek, hogy ott feldolgozásra kerülhessen. Szabályok csak az adatok tárolására vonatkozhatnak.

A második, logikai réteg irányítja magát az alkalmazást, implementálja annak a működéshez szükséges szabályait. Az adat rétegtől egyszerűen elvárja, hogy biztosítson egy általános hozzáférési módot a nyers adatokhoz. Az üzleti réteg dolgozza fel az alsó rétegtől

kapott adatokat, melyeket a feldolgozást követően továbbít a megjelenítési rétegnek. Feladata tehát, hogy kapcsolatot teremtsen a másik két réteggel, amely kapcsolaton keresztül közvetíteni tudja az adatokat. Kezeli a kliensektől érkező szálakat (session) és tehermentesíti is a klienseket. Az alkalmazás szerver segítségével lehetségessé válik a dinamikus oldalgenerálás.

A harmadik réteg a felhasználói interfészt implementálja. Elsődleges feladata, hogy megjelenítse a logikai rétegtől jövő adatokat a kliens készülékének megfelelő módon. Az interfész réteg biztosítja a felhasználó hozzáférését a logikai réteg olyan szolgáltatásaihoz is, mint a lekérdezés vagy az aktualizálás. A felhasználói interfész réteg sohasem kommunikál közvetlenül az adatkezelő réteggel, és nem végez konkrét műveleteket az adatokon. A harmadik rétegben semmilyen üzleti logika nincs, ezért azt mondhatjuk, hogy a kliens vékony. Ez azért is hasznos, mert így a rendszer a klienstől nem követel meg semmit, és emiatt szinte bármilyen gyenge számítógépen is lehet használni az alkalmazást.

Az architektúrában tehát jól elkülöníthető a 3 különböző rendeltetésű szint. A 3 rétegű modellnek köszönhetően teljes mértékben szétválasztható egymástól a három szint fejlesztése. Ez lehetőséget biztosít, hogy a Java nyelvű program írásakor az osztályokat az MVC (Model View Conroller – Adatkezelés Megjelenítés Vezérlés) alapján készítsem el. Ennek az a nagy előnye egyrészt, hogy az architektúra, a program felépítése, és a rendszer működésének logikai struktúrája mind megegyezne, és emiatt talán könnyebbé válna a fejlesztés. Másrészt pedig a jól elkülönülő egységek miatt az újrafelhasználhatóság, és a későbbi rendszerfejlesztés is – például a felhasználói interfész lecserélése – egyszerűbb lenne. Az Abydos fejlesztésekor azonban úgy döntöttem, hogy nem fogom alkalmazni ezt a technikát, mivel szerintem ez valamivel jobban megbonyolította volna a szoftver elkészítését. Emiatt a program hatékonysága nem csökkent, sőt szerintem a kód így is áttekinthető, logikus felépítésű, és könnyen megérthető.

Az alkalmazásomat Java nyelven, elsődlegesen servletek, és JSP (Java Server Pages) technológia használatával készítettem el. A JSP egy klienstől érkező kérés alapján valamilyen szöveges, általában HTML vagy XML formátumú dokumentum dinamikus, szerveroldali előállítására szolgáló technológia. A servlet pedig egy olyan Java objektum, amely HTTP kérést dolgoz fel, és HTTP választ generál, ezzel ugyanúgy a dinamikus tartalom generálás problémáját oldja meg. A JSP szoros kapcsolatban áll a servletekkel, ugyanis a servlet réteg feletti absztrakciós rétegnek tekinthető. Amíg a serveletekben a Java kódba beágyazva

szerepelnek a weboldal kinézetét meghatározó utasítások, addig a JSP oldalak teljesen fordítottan működnek. Ebben az esetben a weblap HTML vagy XML kódjába ágyazott Java kódról beszélhetünk. Az ilyen oldalak futtatásához, szükségünk van egy web szerverre, amit a GlassFish V2 alkalmazás szerver biztosít.

Az Abydos-ban 2 JSP oldal található, és az alkalmazás indításakor ezek jelennek meg. Ezek az oldalak egymásba vannak építve. Az index.jsp a kezdőlap, amivel minden felhasználó találkozik, három részre van bontva, hogy a megjelenés hasonló legyen az Egri Vitézlő Oskola weblapján lévő design-hoz. Az indexSite.jsp oldal pedig arra szolgál, hogy kezdetben a bejelentkezéshez szükséges kezdőoldalt megjelenítse, később pedig a rendszer használata folyamán ez a lap fog változni.

Egy tag bejelentkezésekor a felhasználói név és jelszó megadása után egy autentikációs eljárás kezdődik, ahol a rendszer kapcsolódik az adatbázishoz, és leellenőrzi, hogy van-e ilyen felhasználói név és jelszó pár a felhasználók táblában, és ha igen, akkor megjeleníti a begin.java servlet oldalt. A továbbküldés a következő utasítással történik:

```
response.sendRedirect("http://" + request.getServerName() + ":" +
        Integer.valueOf(request.getServerPort()) +
        "/Abydos/begin?currentUser=" + currentUser);
```

Azért, hogy többféle adatbázis szervert lehessen használni, a szerver nevét, és portját a JSP oldalról kapja meg a servlet, és úgy küldi tovább. A currentUser segítségével folyamatosan számon lehet tartani, hogy ki az aktuális felhasználó.

Az alkalmazás mindig automatikusan felépíti magának a kapcsolatot a JDBC (Java DataBase Connection) adatbázis kezelő programozói interfész segítségével, majd a statement objektum létrehozását követően a program elvégzi a megfelelő adatbázis kezelő SQL utasításokat, és azok feldolgozását, végül létrehozza az eredményül szolgáló servletet, és bontja a kapcsolatot. A következőkben néhány példa látható a servletek szerkezeti felépítésére.

```
response.setContentType("text/html;charset=UTF-8");
request.setCharacterEncoding("UTF-8");
PrintWriter out = response.getWriter();
int currentUser = 0;
if (request.getParameter("currentUser") != null) {
    currentUser = Integer.valueOf(request.getParameter("currentUser"));
}
try {
    Connection con = connections.connect.connect();
    Statement stmt = con.createStatement();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet begin</title>");
        out.println("</head>");
        out.println("<body background=\"hatter.jpg\" bgproperties=fixed>");
        works.buttons.makeButtons(out, con, currentUser);
        out.println("<h2>Üdvözöljük a rendszerben!</h2>");
        out.println("Ön az Abydos-t, az Egri Vitézlő Oskola tag- és eszköznyilvántartó
         alkalmazását használja.");

        out.println("<form name=\"quit\" action=\"exit\" method=\"POST\">");
        out.println("<input type=\"submit\" value=\"Kilépés\" name=\"quit\" />");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
    stmt.close();
    con.close();
} catch (SQLException ex) {
    Logger.getLogger(begin.class.getName()).log(Level.SEVERE, null, ex);
}
```

A most következő kódrészlet azt szemlélteti, hogy módosítási művelet esetén milyen utasítássorozat megy végbe a servletben.

```
/* Jelszó módosítása */
if (request.getParameter("currenttagId") != null &&
      request.getParameter("jelszoModositas") != null) {
   try {
      con.setAutoCommit(false);
      boolean isSuccess = false;
      int tagId = Integer.valueOf(request.getParameter("currenttagId"));
      ResultSet rs = sm.executeQuery("select * from FELHASZNALOK where tagId=" +
       tagId);
      if (rs.next()) {
        if (works.rights.isAdmin(con, currentUser) && currentUser!=tagId &&
              (request.getParameter("jelszo1").equals(rs.getString("jelszo")) ||
       request.getParameter("jelszo1").equals(""))) {
            if ((!request.getParameter("jelszo2").equals("")) &&
       (request.getParameter("jelszo2").equals(request.getParameter("jelszo3")))) {
              stmt.execute("update FELHASZNALOK set jelszo='" +
       request.getParameter("jelszo2") + "' where tagid=" + tagId);
             isSuccess = true;
            }
        }
        else if (request.getParameter("jelszo1").equals(rs.getString("jelszo"))) {
            if ((!request.getParameter("jelszo2").equals("")) &&
       (request.getParameter("jelszo2").equals(request.getParameter("jelszo3")))) {
              stmt.execute("update FELHASZNALOK set jelszo='" +
       request.getParameter("jelszo2") + "' where tagid=" + tagId);
              isSuccess = true;
           }
        }
      }
      rs.close();
      if (isSuccess) {
        out.println("<h2>Sikeres jelszó módosítás!</h2>");
```

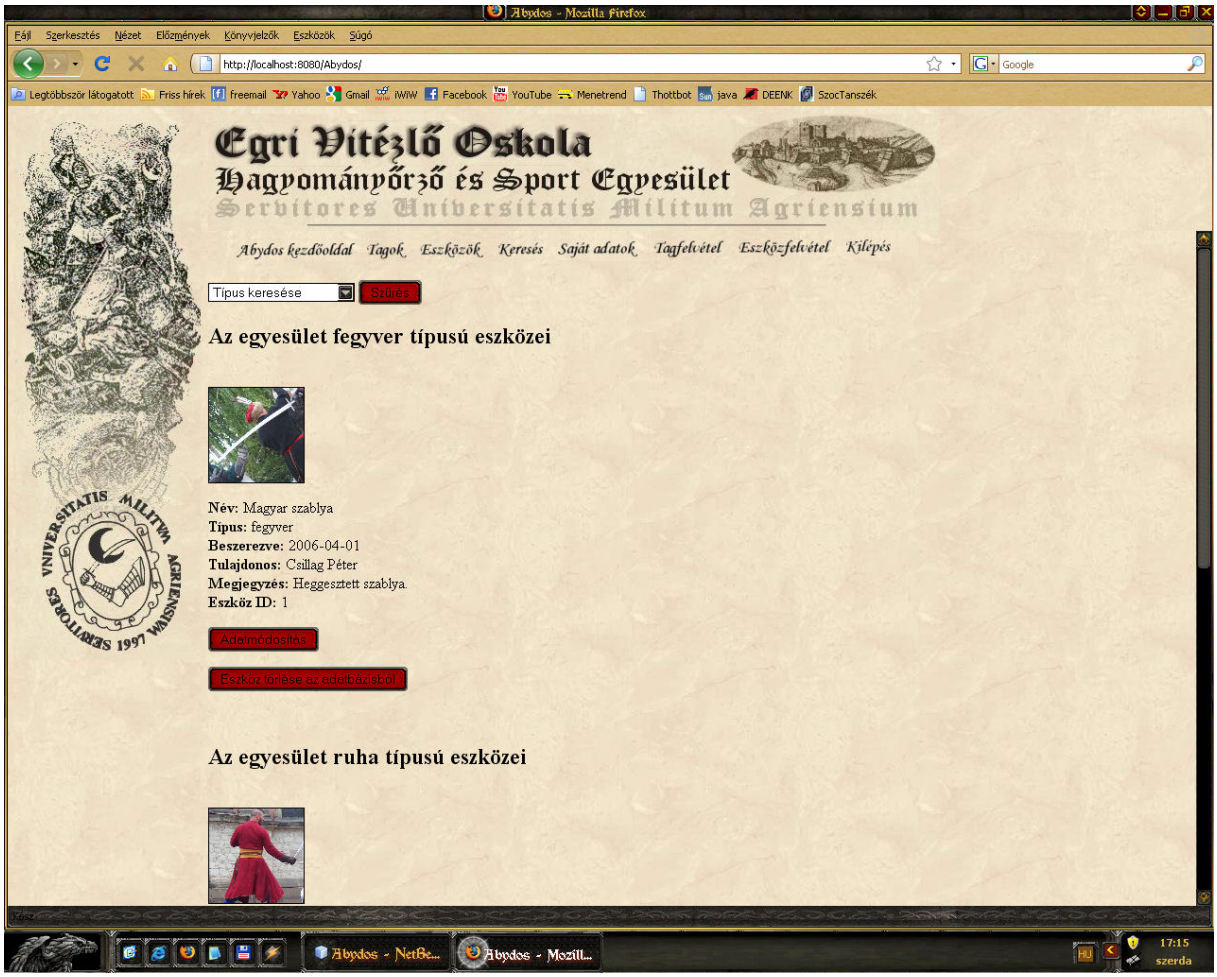
```
            con.commit();
        }
        else {
            out.println("<h2>A módosítás nem sikerült! Kérem próbálja meg újra.</h2>");
            con.rollback();
        }
    } catch (Exception e) {
        out.println("<h2>A módosítás nem sikerült! Kérem próbálja meg újra.</h2>");
        con.rollback();
    } finally {
        con.setAutoCommit(true);
    }
}
```

Felhasználói felület

A felhasználói felület (GUI – Graphic User Interface) tervezésénél nagyon sok dolgot figyelembe kell venni, hogy végül megfelelő eredményt kaphassunk. Egy jó felhasználói felületnek figyelembe kell venni a leendő felhasználó képességeit, határait, valamint az emberek természetét. Az általam elkészített alkalmazás célközönségét – azaz hogy kik fogják használni – nem lehet konkrétan meghatározni, így szinte mindenre fel kellene készíteni a GUI-t. A felhasználói felület tervezésére igen nagy hangsúlyt fektettem, mivel minden, az Abydos-t használó személy ezen keresztül találkozik a rendszerrel. Alapvetően befolyásolja a használhatóságot, és a véleményt is.

A követelményeknek megfelelően, miszerint a megjelenítési stílus ne, vagy csak minimálisan térjen el az Egri Vitézlő Oskola honlapján használatostól, különös figyelmet fordítottam arra, hogy a képek, színek, valamint a menü eleget tegyen ennek az elvárásnak. Ennek megfelelően az oldalak felépítése nagyon hasonló egymáshoz, és amennyire lehet a gombok, és egyéb interakciót kívánó eszközök is hasonló stílust képviselnek. A funkciók használhatóságát megtartva megpróbáltam minimalizálni az alkalmazott technikákat, amit a felhasználóknak használni kell. Így például egy tag adatainak a megadásánál csupán szöveges mezők vannak, nincsenek legördülő listák, vagy egyéb gombok.

Egy ember általánosságban 6-8 dolgot tud egyszerre tárolni a rövid távú memóriájában, ezért a menü például 8 részből áll, és mindegyik csak egy mélységű. A felső menüsort mindig egy külön eljárás írja ki a lap tetejére, így az mindig ugyanúgy néz ki az aktuális felhasználó jogosultságának megfelelően. A rendszer használata nagyon egyszerű, logikus felépítésű, és ha szükséges, értesíti a felhasználót, hogy mi a teendő.



10. ábra – Felhasználói felület (Eszközök listázása)

IX. Összefoglalás

Az Abydos rendszer elkészült, az alkalmazást bármikor üzembe lehet állítani, a szükséges dokumentációk pedig ugyancsak rendelkezésre állnak. A megállapított követelményeknek a szoftver eleget tesz, és bár a tényleges működés még nem kezdődött el, azt hiszem, hogy az Abydos megállja majd a helyét. Természetesen a később felmerülő esetleges hibákat lehet javítani, és igény szerint további fejlesztések elvégzése lehetséges.

Az üzembe helyezést követően még célszerű figyelni egyrészt magát a rendszert, hogy megfelelően működik-e, másrészt pedig a felhasználók reakcióit, és azt, hogy ők milyen véleménnyel vannak az alkalmazás használatáról. Erre a feladatra még érdemes lenne alkalmazni legalább egy informatikust, sőt ha lehet, akkor magát a fejlesztőt. Ezzel a módszerrel nyomon követhető, hogy a felhasználók milyen funkciókat tudnak, vagy nem tudnak használni, vagy hogy milyen szolgáltatásokat szeretnek igénybe venni. Ilyen módon a felhasználók közvetett módon segítenek a rendszer fejlesztésében.

A szoftver fejlesztésének kezdetekor készített terveim alapján – ez tekinthető egyfajta megvalósíthatósági tanulmánynak is – belátható, hogy a tervezett időn belül, minimális költségek mellet sikerült elérni a kitűzött célt, azaz elkészíteni a projektet. Igaz, a követelmények idővel változtak, mégis eredményesnek tekinthető a fejlesztés, hisz a változások az alkalmazás jobbá tételére irányultak.

Az Abydos tehát kész a használatba vételre, megfelelő állapotban van, hogy a megrendelő átvegye a fejlesztőtől, és üzembe helyezze. A szoftver fejlesztése és a dokumentáció ezzel befejeződött.

A Debreceni Egyetemen eltöltött évek alatt, különböző informatikai technológiákat ismertem meg, különböző programnyelvekkel találkoztam és sajátítottam el. A szakdolgozat elkezdése előtt, azt a célt tűztem ki magam elé, hogy megpróbálok betekintést nyerni egy általam még kevésbé ismert webes technológiába az által, hogy elkészítek egy, nagyjából teljes funkcionalitással rendelkező, működő rendszert. A fejlesztés alatt rengeteg új dolgot tanultam meg, és betekintést kaptam egy szoftver fejlesztésének az életciklusaiba, kezdve az elejétől egészen a végéig. Úgy gondolom, amellett, hogy bizonyos fokú tapasztalatra tettem szert, sikerült létrehoznom egy olyan szoftvert, ami akár a valóságban is megállná a helyét. A rendszer elkészítése által szerzett tapasztalatok nagy előnyömre váltak, amit remélem, hogy majd a jövőben is hasznosítani tudok.

X. Irodalomjegyzék

Gábor András – Juhász István (2007): PL/SQL programozás. Budapest, Panem

http://www.vitezlo.hu/ – Bemutatkozás

http://java.sun.com/javase/6/docs/api/

http://java.sun.com/products/javamail/FAQ.html

http://www.tizag.com/htmlT/forms.php

http://www.w3schools.com/TAGS/ref_byfunc.asp

http://wikipedia.org

http://www.google.com

XI. Köszönetnyilvánítás

Köszönetemet szeretném kifejezni azoknak, akik a szakdolgozatom elkészítésében segítségemre voltak.

Témavezetőmnek Kósa Márknak közreműködésért, és az iránymutatásért.

Az Egri Vitézlő Oskola elnökségének, és minden tagjának, akik javaslataikkal segítették a munkámat.