

SZAKDOLGOZAT

Harsányi Sándor

Debrecen
2009

Debreceni Egyetem
Informatika Kar

ALKALMAZÁSFEJLESZTÉS JAVA NYELVEN

*MAGYARORSZÁG TOPOGRÁFIÁJA KÖZÉPISKOLÁSOKNAK
WEBES OKTATÓALKALMAZÁS*

Témavezető:

Dr. habil. Boda István
tanszékvezető, egyetemi docens

Készítette:

Harsányi Sándor
matematika-informatika

Debrecen
2009

Copyright © Harsányi Sándor, 2009
Debreceni Egyetem
Informatika Kar
4010 Debrecen, Hungary

harsanyisandor@freemail.hu

A mű egyéni tanulmányozás céljára szabadon letölthető.
Minden egyéb felhasználás csak a szerző előzetes írásbeli engedélyével történhet.

TARTALOMJEGYZÉK

TARTALOMJEGYZÉK	5
1. BEVEZETÉS	7
2. TELEPÍTÉSI KÉZIKÖNYV	11
2.1. Bevezetés.....	11
2.2. A telepítő CD tartalma.....	11
2.3. Szoftverkövetelmények.....	11
2.4. MySQL 5.0 telepítése.....	12
2.5. Sun Java 2 RE Runtime Environment telepítése.....	13
2.6. Sun Java System Application Server 9.1 telepítése.....	14
2.7. Magyarország topográfiája oktatóalkalmazás telepítése.....	16
2.7.1. Az alkalmazás adatbázisának telepítése.....	16
2.7.2. Erőforrások létrehozása, beállítása.....	18
2.7.3. Az alkalmazás moduljainak telepítése.....	19
3. FELHASZNÁLÓI KÉZIKÖNYV	21
3.1. Bevezetés.....	21
3.2. Topográfiai alapfogalmak.....	21
3.3. Az alkalmazás működése.....	22
3.4. Felhasználói csoportok.....	22
3.5. Az alkalmazás indítása.....	23
3.6. Felhasználói felületek és funkciók.....	24
3.6.1. Általános kezelési leírás	24
Felső menüsor	24
Hibaüzenetek	25
3.6.2. Bejelentkező oldal.....	25
Hivatalos bejelentések.....	26
Bejelentkező űrlap	26
Hibaüzenetek.....	28
3.6.3. Gépház oldal.....	29
3.6.4. Tanári szoba oldal.....	29
Funkciólista.....	29
Tanári funkciók.....	30
Lecke létrehozása.....	30
Gyakorlat létrehozása.....	31
3.6.5. Tanterem oldal.....	32
Topográfia gyakorlatok listája.....	32
3.7. Gyakorlat indítása	33
3.8. Felhasználói felületek.....	34
3.8.1. Nyomógombok.....	34
Indítás gomb	34
Leállítás gomb.....	34
Névjegy gomb	34
Kilépés gomb	34
3.8.2. Dialógusablakok.....	35
Névjegy dialógusablak.....	35
Kilépés megerősítése	35
3.8.3. Beállítások	36
Nyelvi beállítások.....	36
Vizuális beállítások	36
3.8.4. Gyorstippek.....	37
3.8.5. Hátralévő idő.....	37

3.8.6. Névtár.....	38
3.8.7. Vaktérkép.....	39
3.9. Gyakorlat befejezése.....	41
3.10. Kijelentkezés a rendszerből.....	42
4. FEJLESZTŐI KÉZIKÖNYV	43
4.1. Bevezetés.....	43
4.2. Technikai háttér.....	43
4.3. Fejlesztői környezet.....	44
4.4. Az alkalmazás adatbázisának felépítése.....	44
4.4.1. Az authentication tábla.....	46
4.4.2. Az authority tábla.....	46
4.4.3. A session tábla.....	46
4.4.4. A user tábla.....	47
4.4.5. A learnergroup tábla.....	47
4.4.6. A city tábla.....	47
4.4.7. A hidrogeology tábla.....	47
4.4.8. A relief tábla.....	48
4.4.9. A shire tábla.....	48
4.4.10. A lesson tábla.....	48
4.4.11. A task tábla.....	49
4.4.12. A result tábla.....	49
4.5. Az alkalmazás szerkezete.....	49
4.6. A keretalkalmazás felépítése, működése.....	49
4.6.1. JSP oldalak.....	51
4.6.2. A ServiceLoginAuthServlet osztály.....	55
4.6.3. A ServiceLogoutServlet osztály.....	55
4.6.4. A RequestDispatcherServlet osztály.....	55
4.6.5. A DoRequestTag osztály.....	56
4.6.6. A RequestPerformer osztály.....	60
4.6.7. A CommonFunctions osztály.....	60
4.6.8. Az AdminFunctions osztály.....	62
4.6.9. A TeacherFunctions osztály.....	62
4.6.10. A StudentFunctions osztály.....	63
4.6.11. A SOACommonFacadeBean osztály.....	63
4.6.12. A CommonServices osztály.....	64
4.6.13. Az AdminServices osztály.....	65
4.6.14. A TeacherServices osztály.....	65
4.6.15. A StudentServices osztály.....	65
4.6.16. A GatewayServlet osztály.....	66
4.7. A topográfia alkalmazás felépítése, működése.....	66
4.7.1. Saját komponensek.....	69
4.7.2. Hibakeresés.....	73
5. ÖSSZEFOGLALÁS	74
IRODALOMJEGYZÉK	75
FÜGGELÉK.....	77
7.1. Projektek szervezése az Eclipsben.....	79

1. BEVEZETÉS

Felgyorsult világunk egyre nagyobb igényeket és elvárásokat támaszt az oktatással szemben. A technológiai fejlődéssel, az ismeretek bővülésével egyre szaporodnak a tankönyvek oldalapjai is. A követelmények megnövekedése miatt a hagyományos, tanórai keretek között végzet tanítási-tanulási folyamatban egyre kevesebb idő jut a gyakorlásra, a tudás rögzítésére, elmélyítésére. Az órák jelentős részében az új ismeretek átadása és a már megszerzett tudás számonkérése zajlik, míg a gyakorlás többnyire tanórán kívül, házi feladatok formájában történik. Az óvoda és az iskola mellett éppen ezért egyre nagyobb szükség van az otthoni gyakorlásra. Ugyanakkor az otthon végzett munka hatékonyságának biztosítása és eredményességének fenntartása igen körülményes feladat. Ebben lehetnek segítségünkre a különböző számítógépes oktatóprogramok, koordinálva és hatékonyabbá téve az egyénileg, otthon végzett tanulási tevékenységet. Az oktatóprogramokat a tanulási folyamat szerves részének tekintő oktatási forma nem a hagyományos oktatás helyett, hanem azt kiegészítve alkalmazható gyakorlat. Bár ma már szinte mindenki elfogadja a számítástechnikai alkalmazások létjogosultságát az oktatási folyamatban, sőt az oktatás szinte minden területén egyre inkább elterjedőben van a számítógéppel támogatott tanulás, azaz mostanában divatos szóhasználattal az e-learning is, ahhoz hogy az ilyen programok széles körű alkalmazása megvalósulhasson további anyagi beruházásokra, és egyre hatékonyabb, jobb számítógépes oktatóprogramokra van szükség.

A fenti megállapítások alól a földrajztudomány, a földrajz tantárgy oktatása sem kivétel. Napjaink földrajztudományának egyik kiemelkedően fontos ága a topográfia. A topográfia a földfelszín domborzati, vízrajzi és kultúrviszonyaival foglalkozik, különös tekintettel az egyes lakóhelyekre, településekre, városokra, stb. Mindezen viszonyokat a topográfia megadott méretarányban, lekicsinyítve, rajz vagy térkép segítségével, vízszintes vetületen, felülnézetben tünteti fel. Bár a topográfiai ismeretek elsajátítása a földrajz tanulásának nem a lényege, azonban nélkülözhetetlen eszköze annak, ezért a topográfiának a földrajzoktatásban is fontos, központi szerepe van. Ugyanakkor a földrajz tantárgy ismeretanyaga napjainkra jelentős átalakuláson ment keresztül. Az ismeretközpontú oktatás

helyett a képességfejlesztő tanítási folyamatok kerültek előtérbe. Így a tantárgyi ismeretek megszerzése, a Föld természeti és gazdasági jelenségeinek bemutatása, a topográfiai fogalmak tanítása mellett a földrajzoktatás egyik fontos, központi feladata a térszemlélet, a térképeken való tájékozódási képesség, a térképhasználati jártasság fejlesztése lett, különös tekintettel a topográfiai-földrajzi térképekre és a földrajztudományban alkalmazott tematikus ábrázolásokra. Ez a térképismereti-térképhasználati tudás-képesség ugyanakkor különösen jól tanítható-fejleszthető számítógépek segítségével, hiszen a földfelszín elemeinek, tereptárgyainak* pontos, valós viszonyoknak megfelelő ábrázolásában, szemléletes, interaktív digitális térképek készítésében, dinamikus, egyéni igényekre szabható, egyszerűen változtatható, módosítható tájékozódási feladatok, topográfiai tesztfeladatok összeállításában, „játékos” megoldásában és azok gyors kiértékelésében, stb. nagy segítségünkre lehetnek a különféle számítógépes programok.

A középiskolai földrajzoktatásban a tananyag feldolgozásához szükséges topográfiai példákat elsősorban Magyarországról és Európából kell megnevezni. Szakdolgozatom tárgya egy olyan Java nyelven implementált, szolgáltatás alapú, webes oktató-, készségfejlesztő alkalmazás elkészítése, bemutatása és értékelése, amely Magyarország természetföldrajzának és településföldrajzának oktatása során, a hagyományos iskolai oktatást kiegészítve, az ország topográfiájának, topográfiai ismereteinek tanítására, a már megszerzett tudás gyakorlására és elmélyítésére használható. A topográfiai fogalmak megtanulása azonban módszertani szempontból keveset mond, hiszen egy topográfiai fogalmat eltérő mélységben lehet „tudni”. Mászt jelent például ráismerni egy topográfiai fogalomra, mint bejelölni azt egy térkép vázlaton, vaktérképen. Az alkalmazás használata során a tanulás egyszerűen, játékos formában, szemléletes, interaktív digitális térképek segítségével történik. Szemben a hagyományos „asztali” oktatóprogramokkal, ahol a tanulók ugyan egyéni munkatempóban haladva dolgozhatják fel a tananyagot, de a képességek szerinti differenciálásra nincs lehetőség, az elkészített és itt bemutatott alkalmazás – a webes technológiáknak köszönhetően – lehetővé teszi a feladatok személyre szabott kiosztását is. Ennek az elgondolásnak, vagyis a differenciált, egyéni kognitív képességeket figyelembe vevő webes oktatóprogramoknak a létjogosultságát a napjainkban oly népszerű tanulóközpontú oktatás is indokolhatja.

* A Föld felszínén és annak közvetlen közelében lévő természetes alakulatok (pl. domborzat, vízfolyások, növényzet, stb.) és mesterséges létesítmények (pl. települések, utak, vasutak, stb.) összefoglaló neve.

A dolgozat megírását hosszabb ideig tartó tervezési-fejlesztési folyamat előzte meg. A fejlesztés eredményeként létrejött webes alkalmazásban a feladatok jól darabolhatók, eloszthatók több számítógépre, valamint szükség esetén új szolgáltatások fejlesztése is gyorsan, kényelmesen elvégezhető. Az alkalmazás telepíthető (deploy), futtatható kódját és forrását a szakdolgozat CD melléklete tartalmazza.

Ez a dolgozat magában foglalja az alkalmazás felhasználói dokumentációját, a program telepítési folyamatának leírását, telepítési útmutatóját és felhasználói kézikönyvét, valamint tartalmazza az alkalmazás fejlesztői dokumentációját, bemutatja a program felépítését, technikai háttérét, illetve a fejlesztés során alkalmazott irányelveket, megkötéseket.

2. TELEPÍTÉSI KÉZIKÖNYV

2.1. Bevezetés

Ez a fejezet a *Magyarország topográfija középiskolásoknak* webes oktatóalkalmazás telepítési útmutatója. A fejezet magában foglalja az alkalmazás működtetéséhez szükséges szoftvertermékek listáját és azok legfontosabb beállításait, illetve részletesen bemutatja magának az oktatóalkalmazásnak telepítését.

2.2. A telepítő CD tartalma

- ✚ Telepítési útmutató (jelen dokumentum)
(helye: */e-learn/setup/setup.pdf*)
- ✚ MySQL 5.0.41
(helye: */e-learn/setup/environment/mysql-essential-5.0.41-win32.msi*)
- ✚ Sun Java 2 Standard Edition Runtime Environment 5.0
(helye: */e-learn/setup/environment/jre-1_5_0_11-windows-i586-p.exe*)
- ✚ Sun Java System Application Server 9.1_02 (build b04-fcs)
(helye: */e-learn/setup/environment/setup-glassfish-v2ur2-b04.exe*)
- ✚ A program adatbázisát felépítő SQL szkriptek
(helye: */e-learn/setup/dbscripts/*)
- ✚ Az alkalmazás telepíthető moduljai
(helye: */e-learn/setup/appmodules/*)

2.3. Szoftverkövetelmények

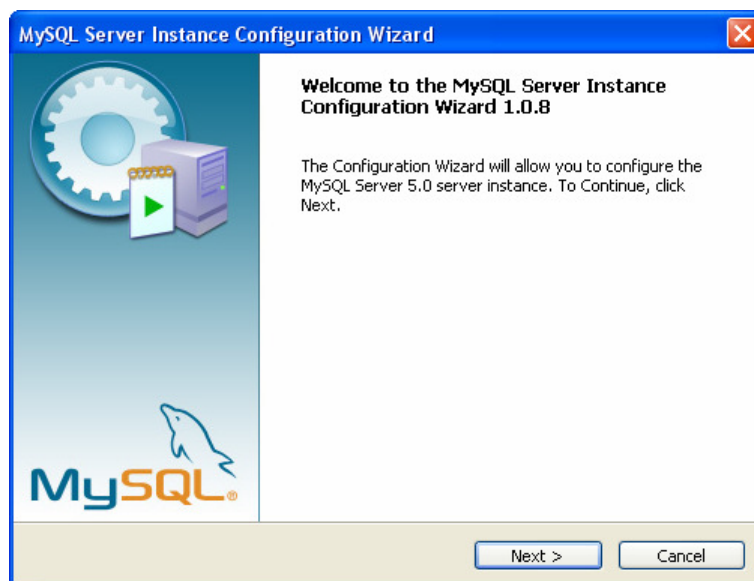
- ✚ Windows XP operációs rendszer
- ✚ MySQL 5.0 SQL-alapú relációs adatbázis-kezelő rendszer
- ✚ Sun Java 2 SE Runtime Environment 5.0 Java futtató környezet
- ✚ Sun Java System Application Server 9.1 vagy Sun GlassFish Enterprise Server v2ur2 alkalmazáserver

2.4. MySQL 5.0 telepítése

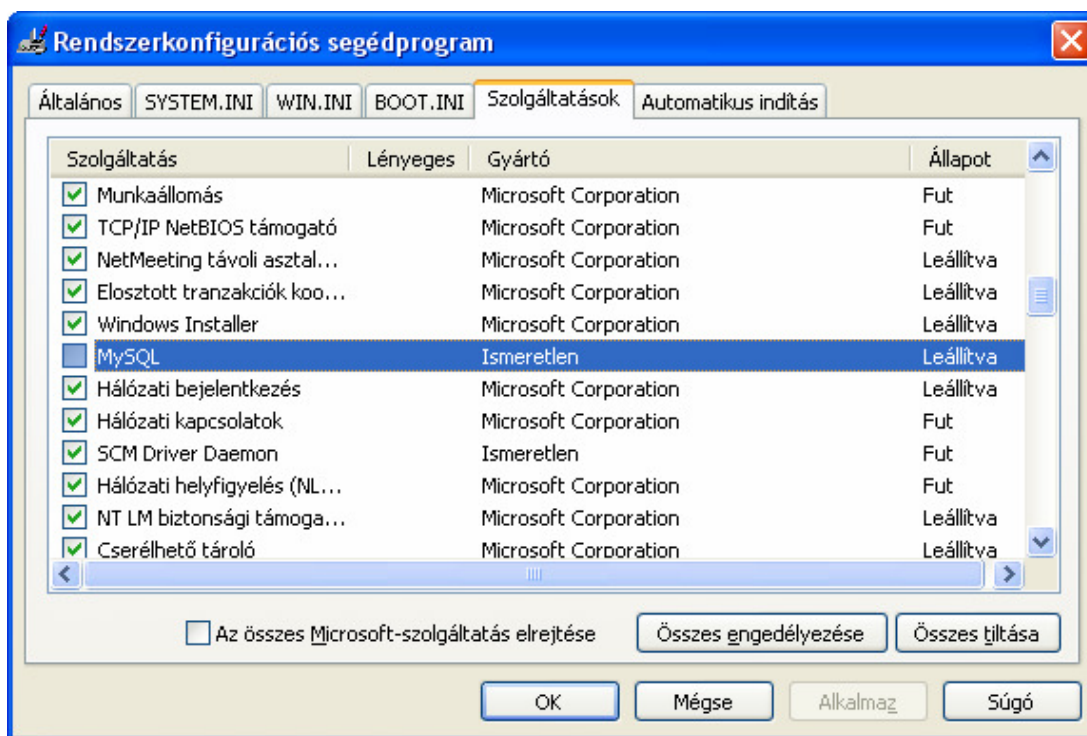
Telepítse a MySQL adatbázis-kezelő rendszert a mellékelt CD-n található *mysql-essential-5.0.41-win32.msi* futtatható állomány segítségével, a telepítés folyamán kövesse a telepítő varázsló utasításait.



A telepítést követően automatikusan elinduló konfigurációs varázsló segítségével, néhány kattintással elvégezheti a rendszer tesztelését, egyedi igényekhez igazítását is. Amennyiben ezt elmulasztaná megtenni, a program könyvtárának *bin* katalógusa alatt található *MySQLInstanceConfig.exe* program segítségével később ismét visszatérhet a varázslóhoz.



Alapértelmezésként a MySQL adatbázis-kezelő szerver az operációs rendszer minden betöltésekor automatikusan elindul. Ha szeretné, hogy a szerver az operációs rendszer indulásakor ne töltődjön be automatikusan, akkor futassa az *msconfig* parancsot, és az elinduló grafikus felületű *Rendszerkonfigurációs segédprogram Szolgáltatások* lapján tiltsa le a *MySQL szolgáltatás* automatikus futtatását. A parancs futtatásához használhatja például a Windows XP operációs rendszer parancsértelmezőjét.



Ezt követően a szerver elindításához használhatja a következő parancsot:

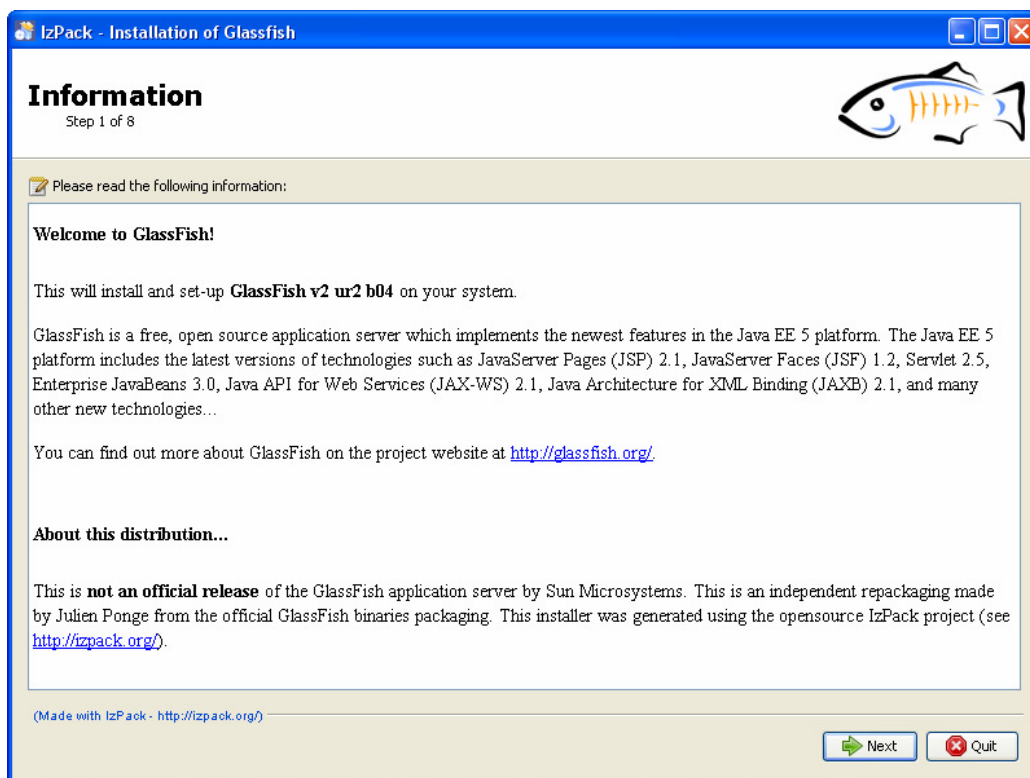
```
C:\Programming Files\MySQL\MySQL Server 5.0.41\bin>mysqld-nt --defaults-file="C:\Programming Files\MySQL\MySQL Server 5.0.41\my.ini"
```

2.5. Sun Java 2 RE Runtime Environment telepítése

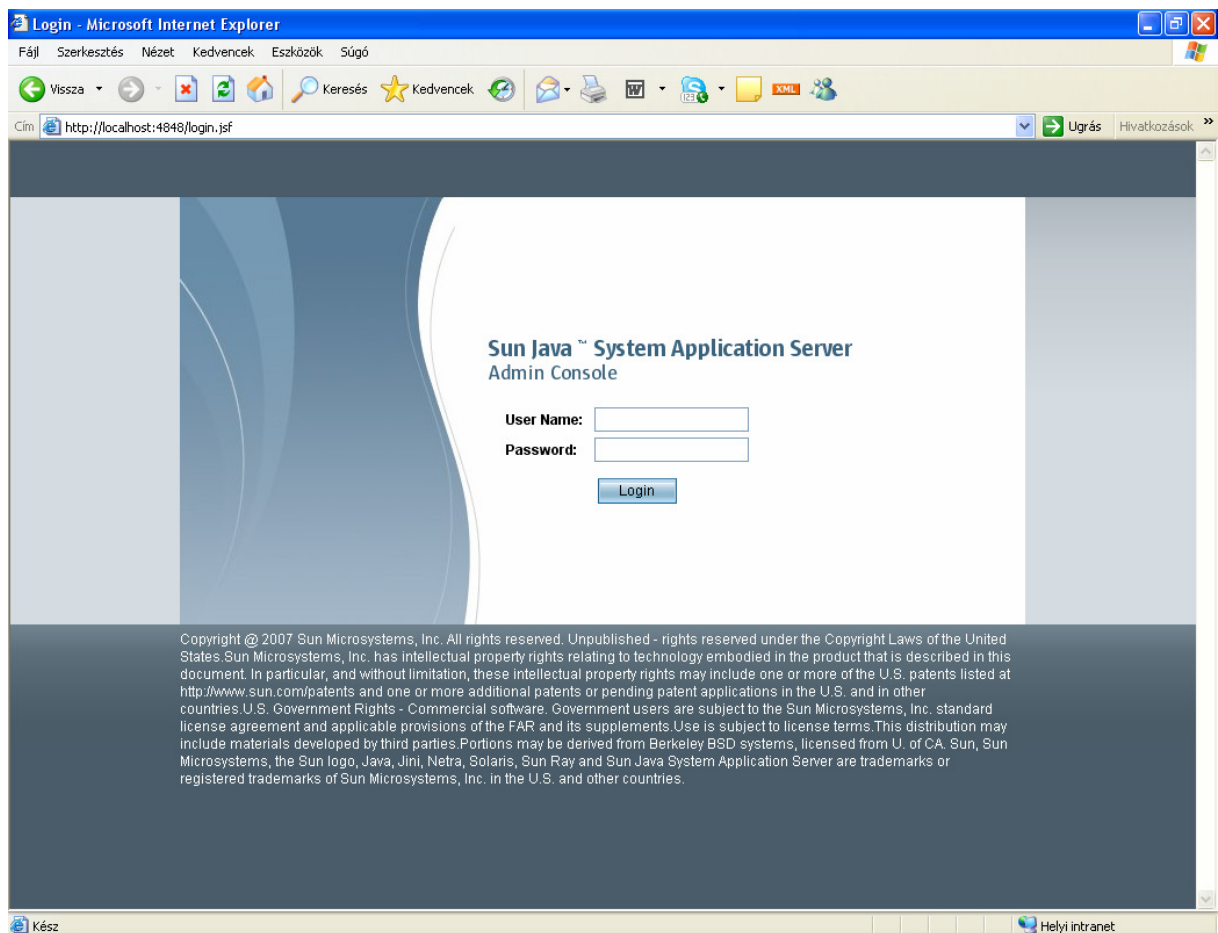
Az alkalmazás futtatásához 5.0 vagy magasabb verziójú Java futtató környezetre van szükség. Az 5.0-ás verzió megtalálható a mellékelt CD-n *jre-1_5_0_11-windows-i586-p.exe* néven, a legfrissebb verzió pedig letölthető a Sun Microsystems hivatalos honlapján, a következő URL címről: <http://java.sun.com/javase/downloads/index.jsp>.

2.6. Sun Java System Application Server 9.1 telepítése

Telepítse az alkalmazásszervert a CD-n található *setup-glassfish-v2ur2-b04.exe* futtatásával, majd kövesse az SJSAS szerver telepítő varázslójának utasításait. A telepítés befejezését követően az alkalmazásszerver automatikusan elindul.



A telepítés sikerességéről könnyen meggyőződhet, ha az Ön által használt böngészőprogram címsorába begépel a <http://localhost:4848> URL címet. Sikeres telepítés esetén a számítógépe sebességétől függően hamarosan meg kell jelennie az alkalmazáserver adminisztrációs oldalának, ahol többek között kényelmes, felhasználóbarát módon végezheti el az alkalmazáserver adminisztrálását, felügyeletét és saját webes alkalmazásai menedzselését.



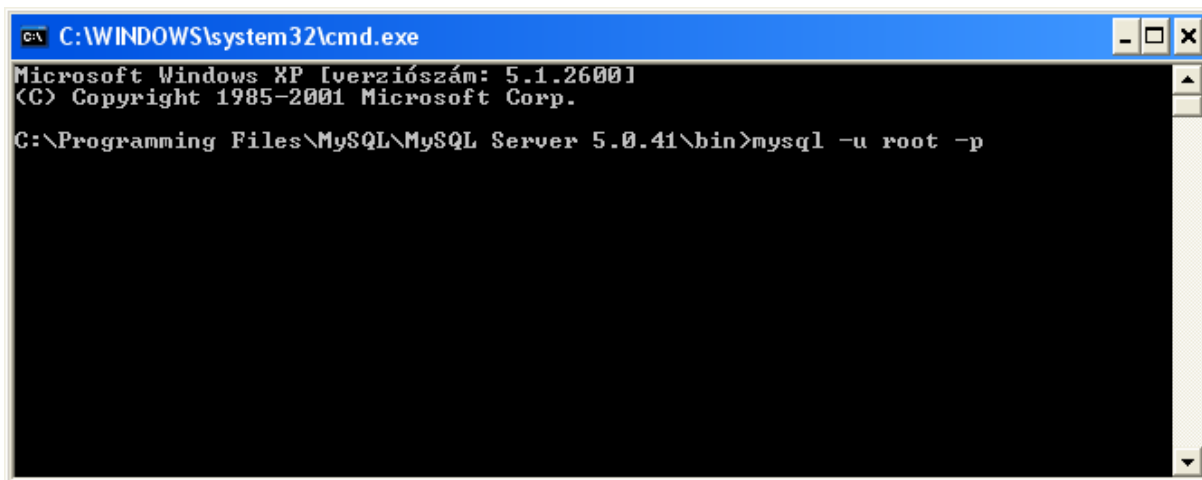
Alapértelmezésként az SJSAS alkalmazáserver –, a MySQL adatbázisszervertől eltérően –, nem indul el automatikusan az operációs rendszer indításakor. A server indításához használhatja a következő parancsot:

```
C:\Programming Files\Java\Glassfish v2 UR2 build b04-fcs\bin>asadmin start-domain domain1
```

2.7. Magyarország topográfiaja oktatóalkalmazás telepítése

2.7.1. Az alkalmazás adatbázisának telepítése

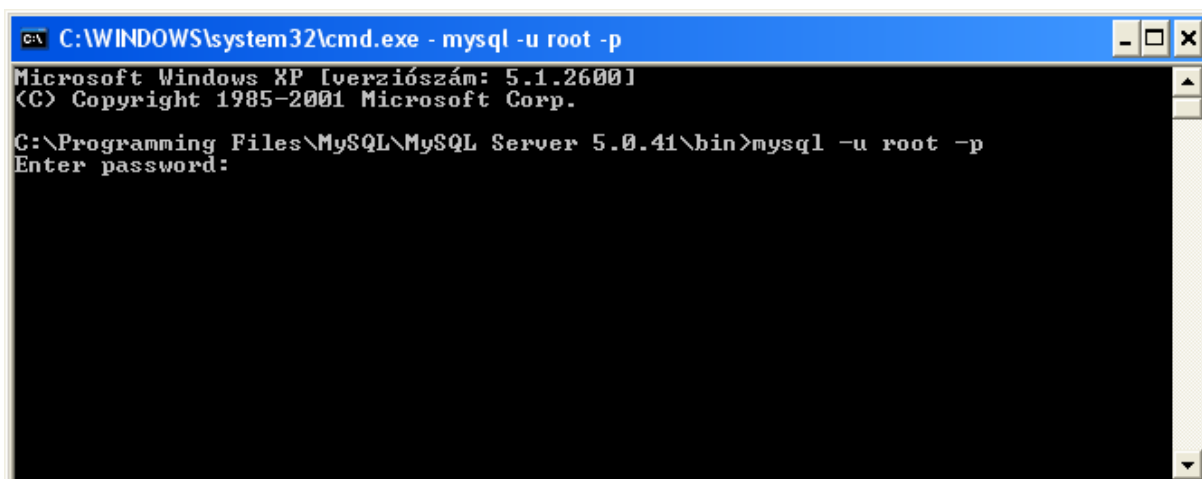
A mellékelt CD */e-learn/setup/dbscripts/* katalógusában található *install.sql*, és *test.sql* állományokat másolja át a MySQL adatbázis-kezelő rendszer *bin* alkönyvtárába, majd futtassa az itt található *mysql.exe* állományt parancssorból a következő paraméterekkel:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [verziószám: 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Programming Files\MySQL\MySQL Server 5.0.41\bin>mysql -u root -p
```

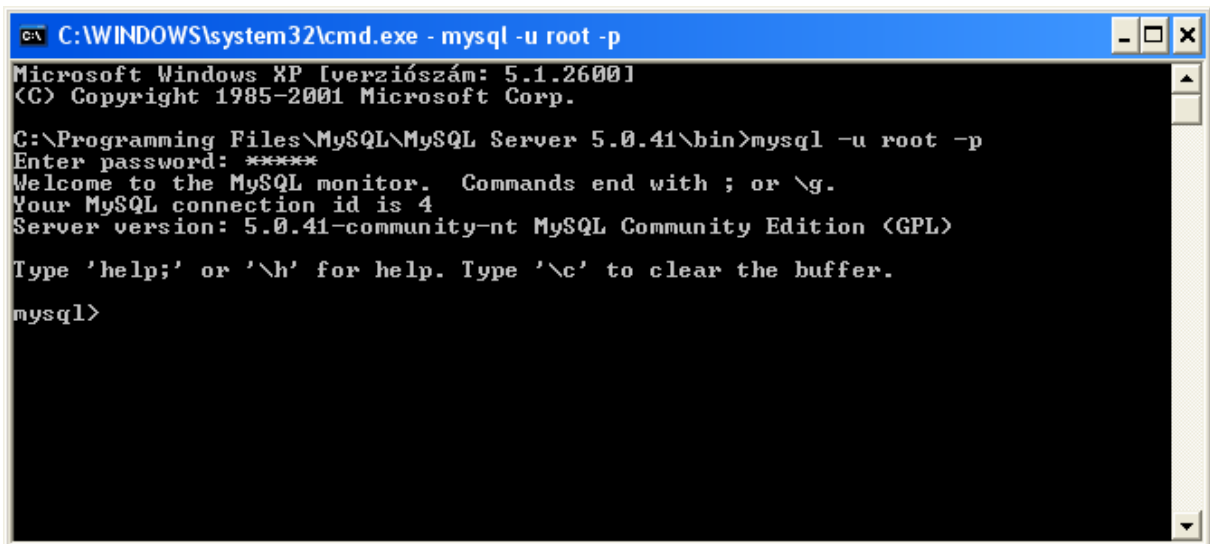
A fent látható *root* azonosítóval rendelkező felhasználó az adatbázis-kezelő rendszer adminisztrátora, akinek joga van a rendszerben tárolt bármely adatbázis eléréséhez, módosításához, új felhasználók létrehozásához és meglévők törléséhez, stb. Ha üt egy ENTER billentyűt, akkor az adatbázis-kezelő rendszer bekéri a felhasználó (*root*) jelszavát. A jelszót a telepítés során Ön adta meg. Most gépelje be!



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
Microsoft Windows XP [verziószám: 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Programming Files\MySQL\MySQL Server 5.0.41\bin>mysql -u root -p
Enter password:
```


Ha mindent jól csinált, akkor megjelenik az adatbázis-kezelő rendszer promptja, jelezve, hogy a rendszer készen áll az utasításai fogadására és feldolgozására.



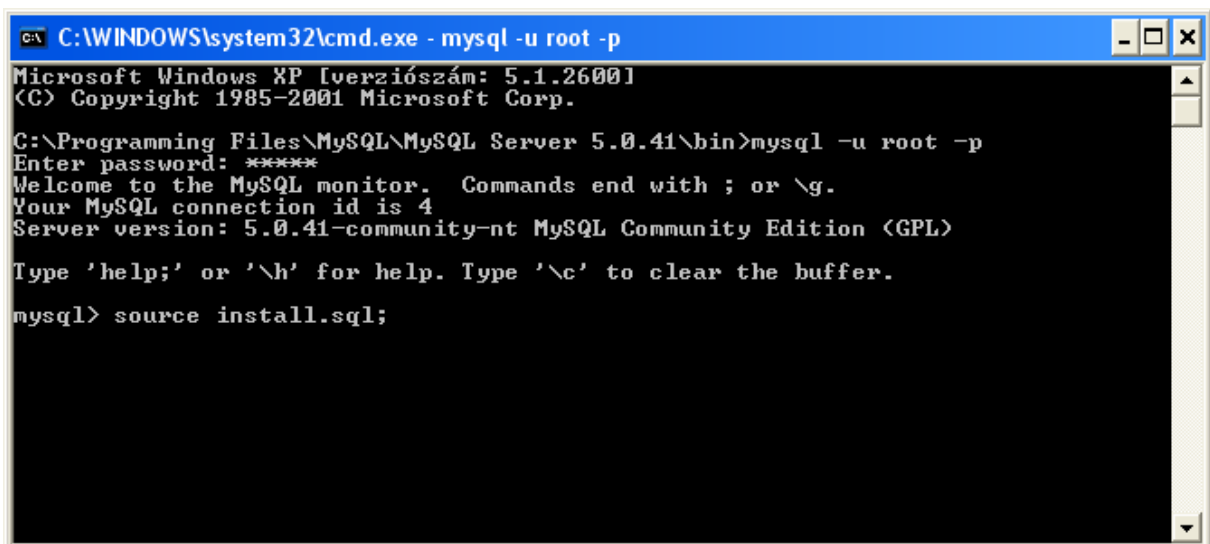
```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
Microsoft Windows XP [verziószám: 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Programming Files\MySQL\MySQL Server 5.0.41\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.41-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Importálja az *install.sql* adatbázis szkriptet a következő parancs kiadásával:



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
Microsoft Windows XP [verziószám: 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

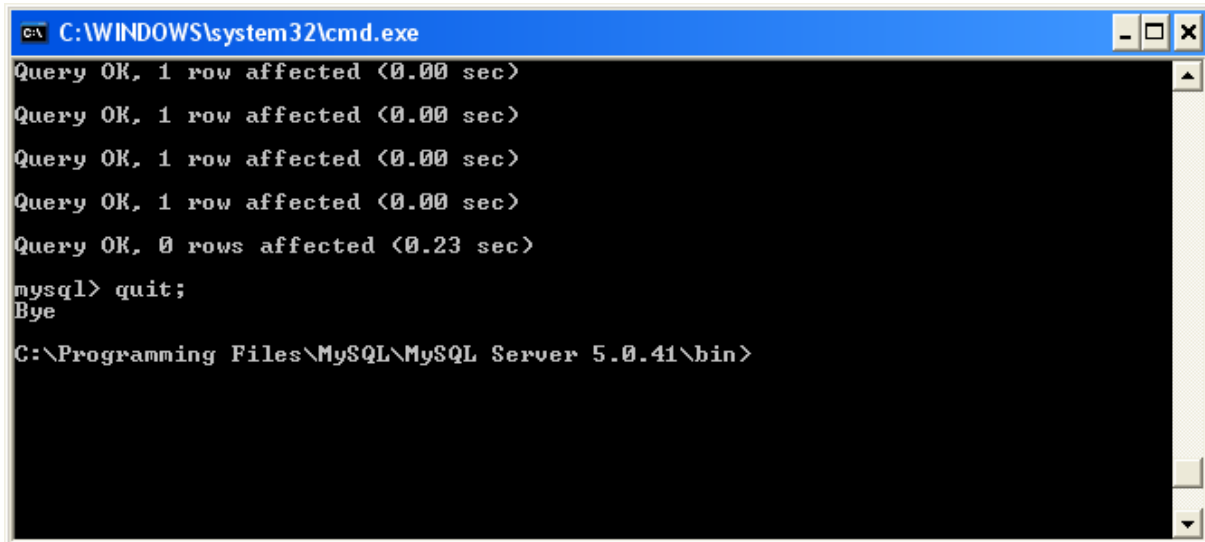
C:\Programming Files\MySQL\MySQL Server 5.0.41\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.41-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> source install.sql;
```

Ha a parancs hibaüzenetek nélkül sikeresen lefutott, és visszakapta az adatbázis-kezelő rendszer promptját, akkor a topográfia oktatóalkalmazás adatbázisa sikeresen importálódott a MySQL adatbázis-kezelő rendszer adatbázisai közé. Ha valamit elrontott, akkor az előbbiekhöz hasonlóan importálja előbb az *uninstall.sql* szkriptet, majd kezdje előlről az *install.sql* állomány importálását.

Ha az importálás sikerült, és nem kapott hibaiüzenetet, akkor az előző lépéseket megismételve a *test.sql* állományra, importálja a tesztadatokat is az adatbázisba, majd a *quit* parancs begépelésével elhagyhatja az adatbázis-kezelő rendszer parancsértelmezőjét.

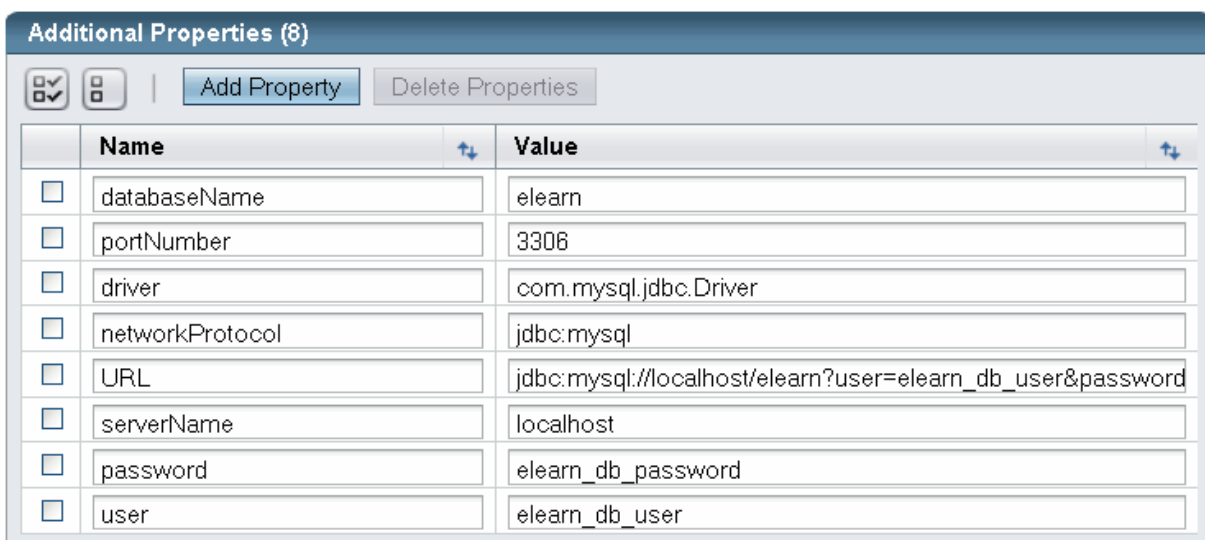


```
C:\WINDOWS\system32\cmd.exe
Query OK, 1 row affected (0.00 sec)
Query OK, 1 row affected (0.00 sec)
Query OK, 1 row affected (0.00 sec)
Query OK, 1 row affected (0.00 sec)
Query OK, 0 rows affected (0.23 sec)
mysql> quit;
Bye
C:\Programming Files\MySQL\MySQL Server 5.0.41\bin>
```

2.7.2. Erőforrások létrehozása, beállítása

Az alkalmazáserver elindítását követően (lásd 2.6. pont) a böngészőprogram címsorába gépelje be a <http://localhost:4848> címet. Ennek hatására megjelenik a SJSAS alkalmazáserver adminisztrációs felületének bejelentkező oldala. Végezze el a bejelentkezést, használja a telepítés során megadott felhasználónevét és jelszavát!

A *Resources/JDBC* menüpont alatt hozzon létre egy JDBC kapcsolat eszköztárat (*Connection Pool*) *ElearnPool* JNDI néven az alábbi paraméterekkel (*Additional Properties*):



	Name	Value
<input type="checkbox"/>	databaseName	elearn
<input type="checkbox"/>	portNumber	3306
<input type="checkbox"/>	driver	com.mysql.jdbc.Driver
<input type="checkbox"/>	networkProtocol	jdbc:mysql
<input type="checkbox"/>	URL	jdbc:mysql://localhost/elearn?user=elearn_db_user&password
<input type="checkbox"/>	serverName	localhost
<input type="checkbox"/>	password	elearn_db_password
<input type="checkbox"/>	user	elearn_db_user

Erőforrástípusként adjon meg *javax.sql.DataSource* adatforrástípust, az adatforrás osztálya pedig legyen *com.mysql.jdbc.jdbc2.optional.MysqlDataSource*.

A tulajdonságok táblázat URL sorában adhatja meg az alkalmazás által használt adatbázis elérési címét. **Figyelem:** a cím az adatbázis eléréséhez szükséges jogosultságokat is tartalmazza. Felhasználónévként adjon meg *elearn_db_user*-t, jelszóként pedig *elearn_db_password*-öt. Ha egyéb felhasználónevet és jelszót szeretne használni, akkor ne felejtse el ezeket az értékeket az adatbázis telepítése előtt az adatbázist felépítő szkriptekben is módosítani.

A *Connection Pool* létrehozását követően készítsen egy JDBC erőforrást (*JDBC Resource*) *java/env/res/jdbc/ElearnDB* JNDI néven, az alábbi beállításokkal:

JNDI Name: * java/env/res/jdbc/ElearnDB

Pool Name: *
Use the [JDBC Connection Pools](#) page to create new pools

Description:

Status: Enabled

2.7.3. Az alkalmazás moduljainak telepítése

Az alkalmazás telepítését (deploy) az *Applications/Enterprise Applications* menüpont alatt végezheti el. Ehhez használja a telepített alkalmazások táblázatának (*Deployed Enterprise Applications*) *Deploy* feliratú nyomógombját. A betöltődő űrlapon (*Deploy Enterprise Applications/Modules*) válassza az Enterprise alkalmazás (*Enterprise Application (.ear)*) típust, majd a *Tallózás* gombra kattintva adja meg az alkalmazás moduljainak fájlrendszeren belüli elérhetőségét. A topográfia oktatóalkalmazás moduljait a mellékelt CD */e-learn/setup/appmodules/* könyvtárában *.ear* kiterjesztésű archív állományokként találhatja meg.

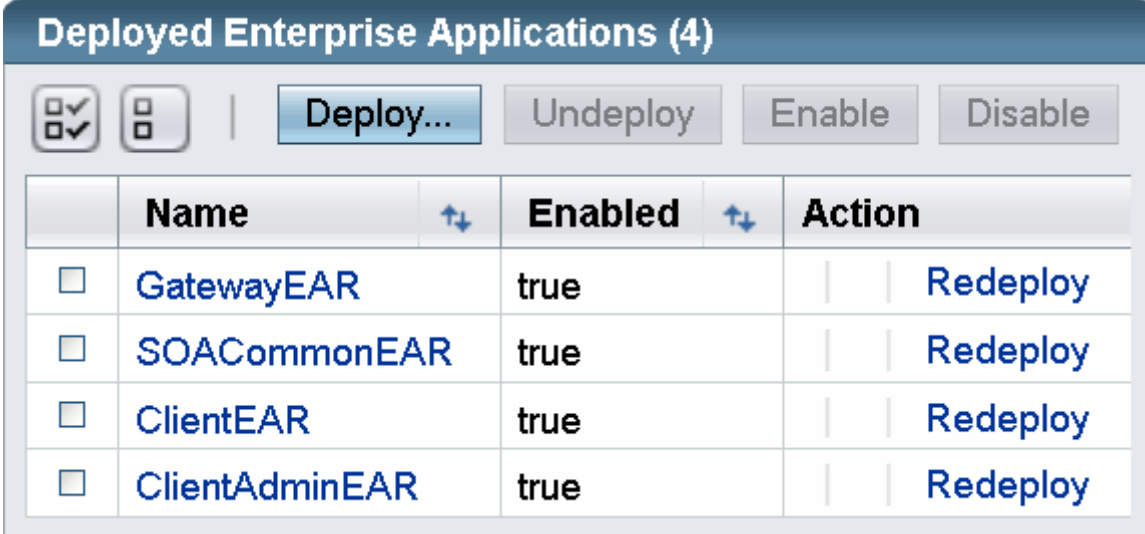
Type:

Location: Packaged file to be uploaded to the server

A modul nevének adja meg az adott modult tartalmazó archív állomány nevét:

Application Name: *

Az CD-n található *.ear* kiterjesztésű archív állományok telepítését követően a képernyőn a következő táblázatot kell látnia:



	Name	Enabled	Action
<input type="checkbox"/>	GatewayEAR	true	Redeploy
<input type="checkbox"/>	SOACCommonEAR	true	Redeploy
<input type="checkbox"/>	ClientEAR	true	Redeploy
<input type="checkbox"/>	ClientAdminEAR	true	Redeploy

Ha befejezte a programmodulok telepítését, akkor lépjen ki az SJSAS alkalmazáserver adminisztrációs felületéről a felület fejlécén található *Logout* vezérlőikon segítségével.

3. FELHASZNÁLÓI KÉZIKÖNYV

3.1. Bevezetés

Ez a fejezet a *Magyarország topográfiája középiskolásoknak* webes topográfia oktatóalkalmazás felhasználói kézikönyve. Az alkalmazás fő célja, hogy egy modern, hálózati technológiákra épülő alternatív megoldást adjon középiskolai tanulmányaikat folytató diákok és tanáraik számára Magyarország topográfiai ismereteinek oktatására, illetve tanulására. Az alkalmazás számítógépes hálózaton keresztül, webes felület segítségével teszi lehetővé, hogy egy adott intézmény, iskola tanárai topográfiai feladatokat, leckéket állítsanak össze, illetve oldassanak meg diákjaikkal. A tanulók az Internet használatával ugyanakkor akár otthonról, házi feladat formájában is elvégezhetik ezen feladatok megoldását. Jelen fejezet az alkalmazás összes felhasználója által közösen használható.

3.2. Topográfiai alapfogalmak

Ennek a dokumentumnak nem célja a topográfia fogalmainak pontos megadása, definiálása, a következő fejezetek és a program használatának megértéséhez azonban feltétlenül szükség van néhány topográfiai alapfogalom tisztázására. Az alábbiakban összefoglaltam a program használatához nélkülözhetetlen alapismereteket.

A Föld felszínén és annak közvetlen közelében lévő természetes alakulatokat, ún. terepalakulatokat (pl. domborzat, vízfolyások, növényzet, stb.) és mesterséges létesítményeket (pl. települések, utak, vasutak, stb.) összefoglaló néven *tereptárgyaknak* vagy *terepelemeknek* nevezzük. A különböző típusú terepelemek neveit összefoglalóan *földrajzi neveknek* nevezzük.

A topográfia feladata a tereptárgyak felmérése és azok térképi ábrázolása. A térképen történő ábrázolás során fel kell tüntetni a terepelemeket és azok földrajzi neveit is. Az ábrázolás során megrajzolt földrajzi neveket *névrajznak* nevezzük. A térképi névrajz egyszerre szöveges információ és rajzi elem.

A *vaktérkép* olyan térkép, amelyből hiányoznak bizonyos információk, az oktatóalkalmazás által összeállítható tájékozási feladatokban használt vaktérképek például olyan térképek, amelyek nem tartalmazzák bizonyos, a tanulók általi meghatározásra szánt tereptárgyak névrajzát, de tartalmazzák azok ábrázolását. A topográfia oktatóalkalmazás célja ezen tereptárgyaknak a beazonosítása egy interaktív vaktérképen.

3.3. Az alkalmazás működése

Az alkalmazás használatával, egy integrált, egységes webes felületen keresztül az alábbi tevékenységek elvégzése lehetséges:

- ▶ Topográfia leckék létrehozása, megtekintése
- ▶ Topográfia gyakorlatok létrehozása, megtekintése
- ▶ Gyakorlatok megoldása

Az alkalmazás ezen feladatok ellátására két egymással szorosan együttműködő alkalmazásmodult biztosít, egy adminisztrációs célú keretalkalmazást és egy topográfiai alkalmazást.

A keretalkalmazás feladata, hogy egy egységes webes felületet adjon a tanárok és a diákok (a továbbiakban felhasználók) számára a fenti feladatok elvégzéséhez, illetve lehetővé tegye az alkalmazás felhasználói számára ezen egységes webes felületre történő ki és bejelentkezést. A keretalkalmazás feladata tehát, hogy a webes felületen keresztül tanárok számára lehetővé tegye leckék és gyakorlatok létrehozását, illetve meglévők lekérdezését, diákok számára pedig gyakorlataik megtekintését és megoldás céljából történő kiválasztását. A keretalkalmazás bemutatásával a 3.5, 3.6, illetve 3.10 fejezetek foglalkoznak.

A topográfiai alkalmazás a keretalkalmazás webes felületéről indítható. Az alkalmazás célja a webes felületen összeállított topográfiai feladatok vizuális, interaktív vaktérkép segítségével történő megjelenítése. A diákok ezen alkalmazás használatával oldják meg a gyakorlataikat. Egy gyakorlat megoldása során a tanulók feladata bizonyos topográfiai információk megkeresése, meghatározása az interaktív vaktérképen. Az alkalmazás leírásával a 3.7-3.9 fejezetek foglalkoznak.

3.4. Felhasználói csoportok

Az alkalmazást három különböző felhasználói csoport használhatja. Ezen csoportokat és az általuk elérhető funkciókat a következő táblázat foglalja össze:

Felhasználói csoport	Funkciók
tanuló	saját gyakorlatok lekérdezése saját gyakorlatok megoldása
tanár	leckék létrehozása saját leckék lekérdezése gyakorlatok létrehozása saját gyakorlatok lekérdezése tanulócsoporthoz lekérdezése
adminisztrátor	adatbázis adminisztráció

Az azonos csoportba tartozó felhasználók azonos jogokkal rendelkeznek, és ennek megfelelően azonos funkciókat érhetnek el. Minden egyes felhasználó csupán egyetlen felhasználói csoportba tartozhat, és minden felhasználót egyetlen felhasználói azonosító azonosít az alkalmazás számára. Új felhasználói azonosítót az adminisztrátor csoportba tartó felhasználók adhatnak. Új felhasználói csoport létrehozására nincs lehetőség.

3.5. Az alkalmazás indítása

Az alkalmazás használatához egy HTML 4.0 szabványt támogató böngészőprogramra van szüksége. Futtassa a programot a böngésző címsorába begépelve a következő URL-t: <http://localhost/admin>

Ha a telepítés során helyesen járt el, akkor az alábbi weblap fog betöltődni:



Figyelem: Ha az alkalmazás telepítésekor nem engedélyezte a JSP oldalak előfordítását (*Precompile JSPs* jelölőnégyzet), akkor az oldalak betöltése az első alkalommal néhány másodpercig is eltarthat.

3.6. Felhasználói felületek és funkciók

3.6.1. Általános kezelési leírás

Felső menüsor

Feladata

Az alkalmazás adatai, a különféle információk és az egyes szolgáltatások, funkciók oldalakra, lapokra bontva, csoportosítva találhatóak meg. A felső menüsor az egyes lapok gyors, közvetlen elérésére szolgál.

Használata

Ha egy adott oldal tartalmát, az azon található információkat meg szeretné tekinteni, akkor kattintson az adott laphoz rendelt menüpontra. Ezt követően a kívánt adatok a szerver sebességétől és az Ön jogosultságaitól függően hamarosan meg fognak jelenni a képernyőn.

Az alábbiakban megtekintheti az egyes felhasználói csoportok által elérhető oldalak listáját. Az oldalak neve minden esetben megegyezik annak a menüpontnak a nevével, amelyre kattintva az adott oldal elérhető. Az egyes oldalakon elérhető, használható felületi elemek és funkciók szerepe a későbbiekben lesz részletezve. (A szakdolgozat idő és tartalmi keretei között nem kerülhetett sor minden oldal implementálására. Amennyiben egy oldal nincs implementálva, azt az adott oldal neve mellett zárójelben feltüntettem.)

Jogosultság nélkül elérhető oldalak

- ▶ Kezdőlap (Bejelentkező oldal)
- ▶ Felhasználói feltételek (nem implementált)
- ▶ Adatkezelési szabályzat (nem implementált)
- ▶ Elérhetőség (nem implementált)

Kezdőlap Felhasználási feltételek Adatkezelési szabályzat Elérhetőség

Az adminisztrátor csoporthoz tartozó felhasználók által elérhető oldalak

- ▶ Gépház
- ▶ Profil (nem implementált)
- ▶ Üzenetek (nem implementált)
- ▶ Fórum (nem implementált)

Gépház Profil Üzenetek Fórum

Kijelentkezés

A tanár csoporthoz tartozó felhasználók által elérhető oldalak

- ▶ Tanári szoba
- ▶ Profil (nem implementált)
- ▶ Üzenetek (nem implementált)
- ▶ Fórum (nem implementált)



A tanuló csoporthoz tartozó felhasználók által elérhető oldalak

- ▶ Tanterem
- ▶ Profil (nem implementált)
- ▶ Üzenetek (nem implementált)
- ▶ Fórum (nem implementált)



Hibaüzenetek

Ha az alkalmazás használatakor, annak működése során valamilyen váratlan hiba következik be, akkor a rendszer erről a váratlan eseményről a következő hibaüzenettel tájékoztatja Önt:

Internal error: The page or service is not accessible. Please try again later!

Ebben az esetben a hiba javítása a rendszeradminisztrátor feladata, próbáljon meg később visszatérni az alkalmazás használatához.

A következő fejezetekben részletesebben olvashat az egyes oldalakon elérhető felületi komponensek és funkciók szerepéről, használatáról.

3.6.2. Bejelentkező oldal

A topográfia oktatóalkalmazás elindítását követően az alkalmazás nyitóoldalát láthatja, amely egyben az alkalmazás bejelentkező oldala is. Ezen az oldalon végezheti el a bejelentkezéssel kapcsolatos teendőit.

Hivatalos bejelentések

Feladata

A *Hivatalos bejelentések* értesítési terület a belső hírek, bejelentések közös felületen történő megjelenítésére szolgál. Itt jelennek meg például a legújabb fejlesztésekről szóló hírek, az új funkciók és szolgáltatások listája, szerepük, használatuk rövid leírása.

Használata

A *Hivatalos bejelentések* felület az oktatóalkalmazás nyitóoldalán helyezkedik el. Ezen a képernyőterületen tulajdonképpen csupán az aktuális bejelentések és hírek címei és azok tartalmának rövid kivonata jelenik meg. A cím, mint hivatkozás azonban hozzáférést biztosít az adott bejelentés teljes szövegéhez, amennyiben a bejelentés teljes szövegére kíváncsi, kattintson annak címsorára.

Képernyőkép



Közvetítő Felhasználói Felületék Hirdetési Szabályzat Elérhetőség

Megújult a vaktérkép szolgáltatás

Felgyorsult világunk egyre nagyobb igényeket és elvárásokat támaszt az oktatással szemben. A technológiai fejlődéssel, az ismeretek bővülésével egyre szaporodnak a tankönyvek oldallapjai. A követelmények növekedése miatt a hagyományos, tanórai keretek között végzet tanítási-tanulási folyamatban egyre kevesebb idő jut a gyakorlásra, a tudás rögzítésére, elmélyítésére. Az órák jelentős részében az új ismeretek átadása és a már megszerzett tudás számonkérése zajlik, míg a gyakorlás többnyire tanórán kívül, házi feladatok formájában történik. Az iskolai oktatás mellett ezért egyre nagyobb szükség van az otthoni gyakorlásra.

Ugyanakkor az otthon végzett munka hatékonyságának biztosítása és eredményességének fenntartása igen körülményes feladat. Ebben lehet segítségünkre a most megújult "vaktérkép" oktatóprogram, amely koordinálja és hatékonyabbá teszi az egyénileg, otthon végzett tanulási tevékenységet.

Be
Az
Je
Elfe
Nem

© Copyright by Developer and Publisher Services (DPS) | Design by Gábor Horváth

Bejelentkező űrlap

Feladata

Egy adott felhasználó bejelentkezéséhez, a bejelentkezéshez szükséges jogosultságok ellenőrzéséhez szükséges adatok megadására és azok elküldésére szolgál.

Használata

Minden felhasználót, így Önt is, egy felhasználói azonosító és egy jelszó azonosít. A bejelentkezés az alábbi űrlap kitöltésével, és annak elküldésével történik. Az űrlapon két mező értékét kell megadnia, az első mezőbe a felhasználói azonosítóját, míg a másodba az adott azonosítóhoz tartozó, azt megerősítő jelszavát kell begépelnie. Mivel a jelszó bizalmas, személyes információ, ezért a mező kitöltésekor a bevitt karakterek helyett úgynevezett helyettesítő karaktereket fog látni.

Az azonosító adatok begépelését követően az ENTER billentyű leütésével, vagy az egér baloldali gombjával a *Bejelentkezés* gombra kattintva küldheti el adatait.

Bejelentkezés

Ha Ön rendszeradminisztrátor, akkor az alkalmazás és a teszt adatbázis telepítését követően, a belépéshez a következő tesztadatokat, azonosítókat és jelszavakat használhatja:

azonosító: admin	jelszó: passwd
azonosító: teacher1	jelszó: passwd
azonosító: teacher2	jelszó: passwd
azonosító: student1	jelszó: passwd
azonosító: student2	jelszó: passwd

Figyelem: Vigyázzon arra, hogy a felhasználónév és a jelszó „case sensitive” adatok, azaz a rendszer mind a felhasználónevek, mind a jelszavak esetében megkülönbözteti a kis- és a nagybetűket!

Képernyőkép

Bejelentkezés

Azonosító:

Jelszó:

Bejelentkezés

[Elfelejtetted a jelszavad?](#)

[Nem vagy még tag?](#)

Hibaüzenetek

A jogosultságok ellenőrzése során bekövetkező bármely hiba sikertelenné teszi a bejelentkezést. Amennyiben a probléma a Ön saját hibájából következett be, például az űrlap kitöltése során elírta a felhasználónevét vagy jelszavát, illetve hiányosan adta meg az adatokat, akkor a program a kérés elutasításáról és a hiba okáról a következő hibaüzenetek valamelyikével tájékoztatja Önt:

Érvénytelen felhasználónév! Próbálja újra!

Ezt a hibaüzenetet akkor kapja, ha a bejelentkező űrlap kitöltése során elírta vagy nem adta meg a felhasználónevét.

A megadott jelszó érvénytelen! Próbálja újra!

Ezt a hibaüzenetet akkor kapja, ha az azonosító adatai bevitele során elrontotta jelszava megadását.

Képernyőkép

The image shows two screenshots of a login page titled "Bejelentkezés". Each screenshot has a light beige background. The top section contains the title "Bejelentkezés" and two input fields: "Azonosító:" and "Jelszó:". Below the fields is a dark button labeled "Bejelentkezés".

The left screenshot shows an error message in red text: "Érvénytelen felhasználónév! Próbálja újra!". Below the message is a link: "Elfelejtetted a jelszavad? Nem vagy még tag?".

The right screenshot shows an error message in red text: "A megadott jelszó érvénytelen! Próbálja újra!". Below the message is a link: "Elfelejtetted a jelszavad? Nem vagy még tag?".

Ha Ön adminisztrátor/tanár/tanuló, akkor sikeres bejelentkezést követően a *Gépház/Tanári szoba/Tanterem* oldalra kerül, ahol lehetősége van az Ön jogosultságainak megfelelő funkciók elérésére.

3.6.3. Gépház oldal

Ha Ön adminisztrátor, akkor az alkalmazásba történő bejelentkezését követően, vagy a *Gépház* menüpont kiválasztása után a *Gépház* oldalra kerül. A *Gépház* oldal jelenleg nem tartalmaz egyetlen elérhető funkciót sem.



3.6.4. Tanári szoba oldal

Ha Ön tanár, akkor az oktatóprogramba való bejelentkezését követően, vagy a *Tanári szoba* menüpont kiválasztása után a *Tanári szoba* oldalra kerül. Az alábbiakban a *Tanári szoba* oldalon megtalálható felületi elemekről és funkciókról olvashat.



Funkciólista

Feladata

A tanári funkciók eléréséhez használható.

Használata

A tanári funkciók listája a *Tanári szoba* lap jobb oldalán található. Egy adott tanári funkció eléréséhez kattintson az egér bal oldali nyomógombjával az adott funkciónak megfelelő listaelemre.

Képernyőkép



Tanári funkciók

Lecke létrehozása

Ezen funkció segítségével egy új leckét hozhat létre. A funkció a következő beviteli elemeket tartalmazza:

- ▶ **Lecke neve:** A létrehozandó lecke megnevezése
- ▶ **Megoldásra ajánlott időkeret:** A lecke megoldására, azaz leckében megadott tereptárgyak vaktérképen történő elhelyezésére ajánlott időintervallum hossza.

Lecke neve:

Megoldásra ajánlott időkeret:

- ▶ **Települések:** Azon települések listáját adhatja meg a segítségével, amelyeket a leckében szeretne számonkérni. A települések kijelölése az adott település neve mellett található kijelölő doboz () segítségével történhet.

Települések	
1. Budapest	<input type="checkbox"/>
2. Debrecen	<input type="checkbox"/>
3. Eger	<input type="checkbox"/>
4. Győr	<input type="checkbox"/>
5. Kecskemét	<input type="checkbox"/>
6. Mátészalka	<input type="checkbox"/>
7. Miskolc	<input type="checkbox"/>
8. Mohács	<input type="checkbox"/>
9. Nyíregyháza	<input type="checkbox"/>
10. Pécs	<input type="checkbox"/>
11. Sopron	<input type="checkbox"/>
12. Szeged	<input type="checkbox"/>
13. Szolnok	<input type="checkbox"/>
14. Veszprém	<input type="checkbox"/>
15. Záhony	<input type="checkbox"/>
16. Zalaegerszeg	<input type="checkbox"/>

- ▶ **Domborzati elemek:** Azon domborzati elemek listáját adhatja meg a segítségével, amelyeket a leckében szeretne számonkérni. Az elemek kijelölése az adott elem neve mellett található kijelölő doboz segítségével történhet.

Domborzati elemek	
1. Bükk	<input checked="" type="checkbox"/>
2. Mecsek	<input type="checkbox"/>

- ▶ **Vízrajzi elemek:** Azon folyók, tavak, stb. listáját adhatja meg a segítségével, amelyeket a leckében szeretne számonkérni. A vízrajzi elemek kijelölése az adott elem neve mellett található kijelölő doboz segítségével történhet.

Vízrajzi elemek	
1. Duna	<input type="checkbox"/>
2. Szamos	<input type="checkbox"/>
3. Tisza	<input type="checkbox"/>

- ▶ **Megyék:** Azon megyék listáját adhatja meg a segítségével, amelyeket a leckében szeretne számonkérni. A megyék kijelölése az adott megyenév mellett található kijelölő doboz segítségével történhet.

Megyék

1. Borsod-Abaúj-Zemplén <input type="checkbox"/>	2. Hajdú-Bihar megye <input type="checkbox"/>
megye	
3. Szabolcs-Szatmár- <input type="checkbox"/>	
Bereg megye	

- ▶ **Megjegyzés a leckével kapcsolatban:** Ebben a mezőben a leckével kapcsolatos megjegyzéseket adhat meg.

Megjegyzés a leckével kapcsolatban:

Gyakorlat létrehozása

Ezen funkció segítségével egy új gyakorlatot hozhat létre. A funkció a következő beviteli elemeket tartalmazza:

- ▶ **Dátum:** A gyakorlat megoldására javasolt ütemtervet állíthatja be vele.

Dátum: -tól -ig

A dátum megadásához használhatja a beviteli mezők mellett található ikont is. Ha erre az ikonra kattint, akkor a jobboldali képen látható naptár segítségével választhatja ki a megfelelő dátumot. Az egyes hónapok között az ◀ és ▶ ikonokkal navigálhat.

◀ Április 2009 ▶						
H	K	Sz	Cs	P	Sz	V
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

- ▶ **Leckék:** Ebből a listából választhatja ki az adott gyakorlat során gyakoroltatni kívánt leckét. Egy gyakorlat során egyetlen lecke gyakoroltatható. A lecke kiválasztásához az adott lecke mellett található rádiógombot használhatja.

Leckék (gyakorló feladatok)

- [lecke: lesson_1 name - description: lesson \(1\)](#)
- [lecke: lesson_2 name - description: lesson \(2\)](#)

- ▶ **Tanulócsoporthok:** Ebből a listából választhatja ki azt a tanulócsoporthot, amellyel az adott leckét meg szeretné oldatni. A csoport kiválasztásához az adott csoport mellett található rádiógombot használhatja.

Tanulócsoporthok

- csoport: learnergroup 1 name - description: learnergroup (1)
- csoport: learnergroup 2 name - description: learnergroup (2)
- csoport: learnergroup 3 name - description: learnergroup (3)

3.6.5. Tanterem oldal

Ha Ön tanuló, akkor az oktatóprogramba való bejelentkezését követően, vagy a *Tanterem* menüpont kiválasztása után a *Tanterem* oldalra kerül. Ezen az oldalon található meg az Ön számára kitűzött topográfia gyakorlófeladatokat.

Tanterem Profil Üzenetek Fórum Oldaltérkép Kijelentkezés

Topográfia gyakorlatok listája

Feladata

A *Topográfia gyakorlatok listája* az aktuális gyakorló feladatainak elérésére, egy adott gyakorlat megoldás céljából történő kiválasztására szolgál.

Használata

Ez a lista azon topográfia gyakorlatok listája, amelyeknek az elvégzését korábban egy oktató előírta az Ön számára. Ezen listából választhatja ki, hogy az aktuális gyakorlatok közül melyiknek a megoldását szeretné megcsinálni. A kiválasztáshoz kattintson a lista adott elemére az egér baloldali nyomógombjával.

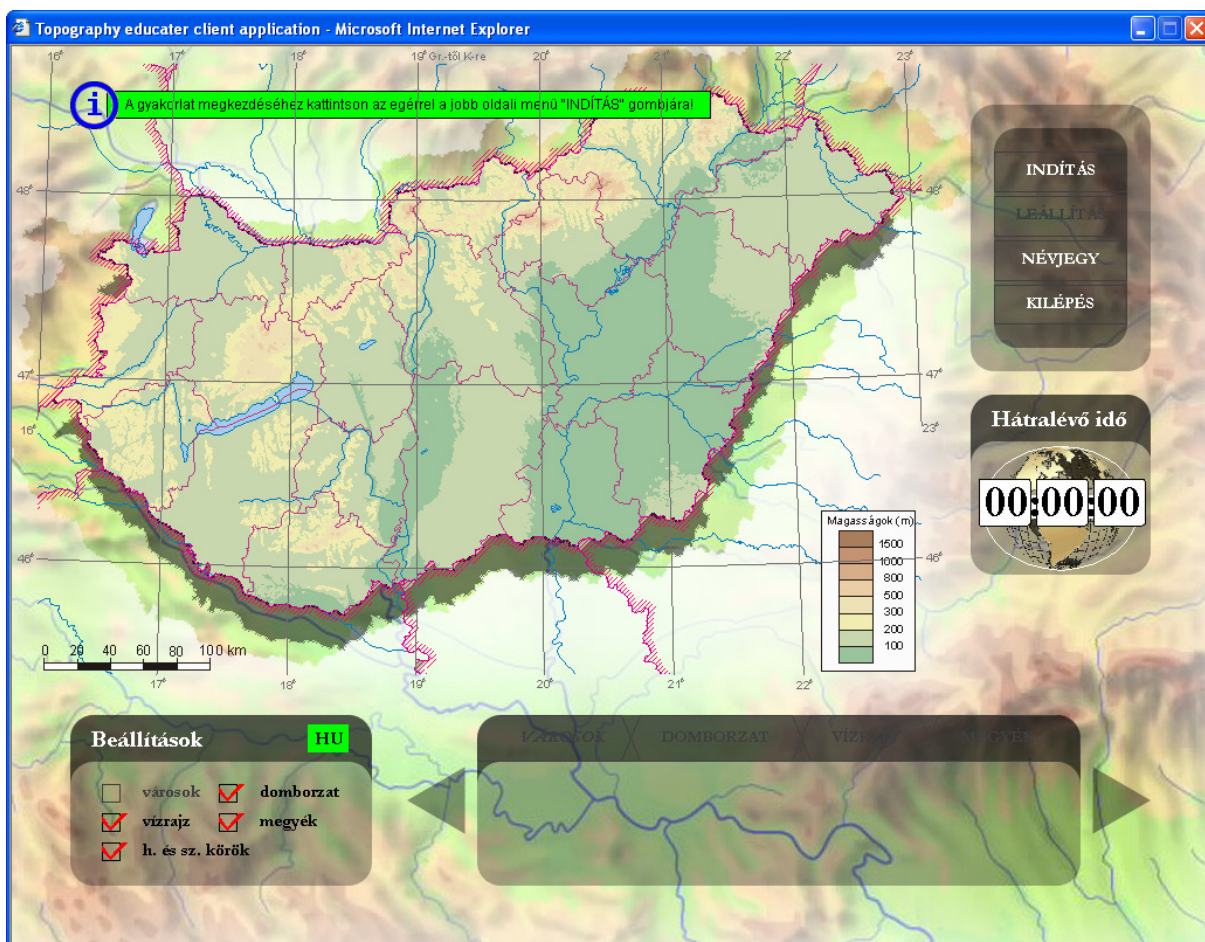
Képernyőkép

Topográfia gyakorlatok

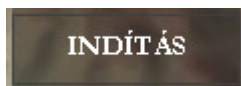
1. feladat: lesson 1 name , learnergroup 1 name (2008-01-01 - 2009-12-31)
2. feladat: lesson 1 name , learnergroup 2 name (2008-01-01 - 2009-12-31)
3. feladat: lesson 2 name , learnergroup 2 name (2008-01-01 - 2009-12-31)

3.7. Gyakorlat indítása

Egy adott gyakorlófeladat kiválasztásához kattintson a *Tanterem* oldal *Topográfia gyakorlatok* listáján az adott gyakorlatra, mint listaelemre. Ennek hatására egy új böngészőablakban az alábbi alkalmazás fog betöltődni:



A kiválasztott gyakorlat megoldásának elkezdéséhez kattintson az elindított gyakorlóalkalmazás *Indítás* feliratú nyomógombjára.

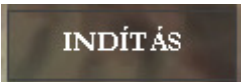


3.8. Felhasználói felületek

3.8.1. Nyomógombok

Indítás gomb

Az *Indítás* feliratú nyomógomb az alkalmazás képernyőképének jobb felső sarkában található. Ezt a gombot használhatja az Ön által kiválasztott gyakorlat megoldásának elindításához.

A képen egy sötét, téglalap alakú gomb látható, amelyen a szó "INDÍTÁS" fehér, nagybetűs betűkkel van felírva.

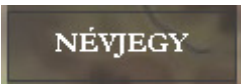
Leállítás gomb

A *Leállítás* feliratú nyomógomb az alkalmazás képernyőképének jobb felső sarkában található. Ezt a gombot használhatja az Ön által kiválasztott gyakorlat megoldásának befejezéséhez.

A képen egy sötét, téglalap alakú gomb látható, amelyen a szó "LEÁLLÍTÁS" fehér, nagybetűs betűkkel van felírva.


Névjegy gomb

A *Névjegy* feliratú nyomógomb az alkalmazás képernyőképének jobb felső sarkában található. Ezt a gombot használhatja az alkalmazás névjegyének megtekintéséhez. A névjegy jelenleg csak a készítőről tartalmaz információkat.

A képen egy sötét, téglalap alakú gomb látható, amelyen a szó "NÉVJEGY" fehér, nagybetűs betűkkel van felírva.

Kilépés gomb

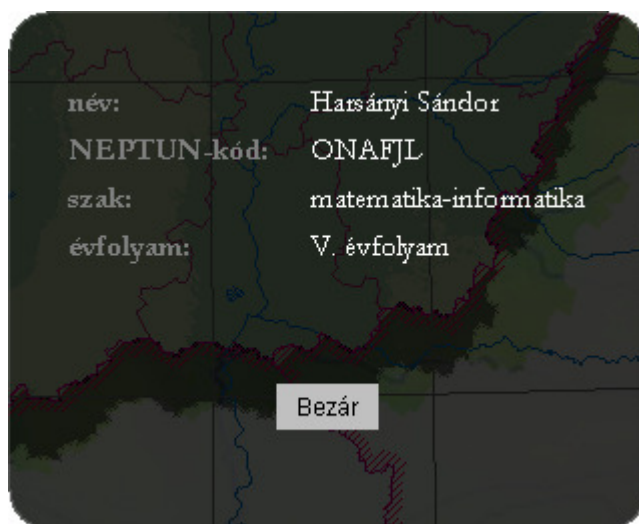
A *Kilépés* feliratú nyomógomb az alkalmazás képernyőképének jobb felső sarkában található. Ezt a gombot használhatja a gyakorló alkalmazás működésének befejezéséhez.

A képen egy sötét, téglalap alakú gomb látható, amelyen a szó "KILÉPÉS" fehér, nagybetűs betűkkel van felírva.

3.8.2. Dialógusablakok

Névjegy dialógusablak

A *Névjegy* nyomógomb hatására megjelenő dialógusablak. Ezen a dialógusablakon olvashatja el az alkalmazás névjegyét. A névjegy jelenleg csak a készítőről tartalmaz információkat. A *Névjegy* dialógusablak bezárásához használja a dialógusablak *Bezárás* feliratú gombját.



Kilépés megerősítése

Ez a dialógusablak azután jelenik meg a képernyőn, miután a gyakorló alkalmazás *Kilépés* gombjára kattintott. Ekkor ugyanis a programnak szüksége van az Ön kilépési szándékának megerősítésére.

A kilépés megerősítéséhez használja a dialógusablak *Igen* feliratú nyomógombját. Amennyiben mégsem szeretne kilépni a programból, válassza a *Nem* feliratú nyomógombot.



3.8.3. Beállítások

A beállítások az alkalmazás képernyőképének bal alsó sarkában található panelen végezhetőek el. A *Beállítások* panel az oktató-, gyakorlóalkalmazás legfontosabb beállításainak gyors elérésére szolgál. Ezen a területen végezheti el az alkalmazás nyelvi beállításait, illetve bizonyos vizuális beállításokat.



Nyelvi beállítások

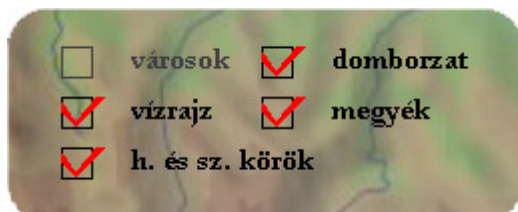
Az alkalmazás jelenleg kétféle nyelvi beállítást támogat, a magyar és az angol nyelvi beállításokat. A gyakorlatok alapértelmezésként magyar nyelvi beállításokkal indulnak. Amennyiben szeretné megváltoztatni ezt az alapértelmezést, a nyelvi beállítások módosításához kattintson a következő ikonra:

HU

A nyelvi beállítás megváltoztatását követően az alkalmazás szöveges erőforrásai az Ön által kiválasztott nyelven jelennek meg a képernyőn.

Vizuális beállítások

A vizuális beállítások az oktatóalkalmazás vaktérképére, illetve annak terepelemeire vonatkozó beállítások. Az egyes beállítások engedélyezését, illetve letiltását egy-egy jelölőnégyzet kiválasztásával, illetve kiválasztásának megszüntetésével adhatja meg.



Mind a nyelvi, mind pedig a vizuális beállítások a gyakorlatvégzés folyamán bármikor megváltoztathatóak

3.8.4. Gyorstippek

Feladata

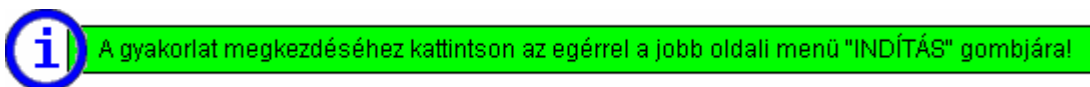
A gyakorlat megoldása során az aktuálisan elvégezhető feladatiról minden esetben értesítést kap a képernyő bal felső sarkában, a vaktérkép fölött megjelenő gyorsstippek formájában. A gyorsstippek segítségével tájékozódhat az éppen aktuálisan elvégezhető tevékenységeiről, illetve arról, hogy a gyakorlat megoldásában éppen hol tart.

Használata

Amikor egy gyorsstipp aktuálissá válik, akkor a tipp szövege automatikusan megjelenik a képernyőn, majd néhány másodpercet követően eltűnik onnan. Ha a tipp szövegét elmulasztotta elolvasni, akkor a szöveg ismételt megjelenítéséhez húzza az egérkurzort a következő ikonra:



Képernyőkép



3.8.5. Hátralévő idő

Feladata

Amikor egy tanár létrehoz egy leckét, akkor az adott leckében található topográfiai feladatok megoldására ad egy ajánlott időkeretet. Ez az időintervallum csupán tájékoztató jellegű, a gyakorlat az időkeret túllépését követően is folytatható és befejezhető.

Képernyőkép

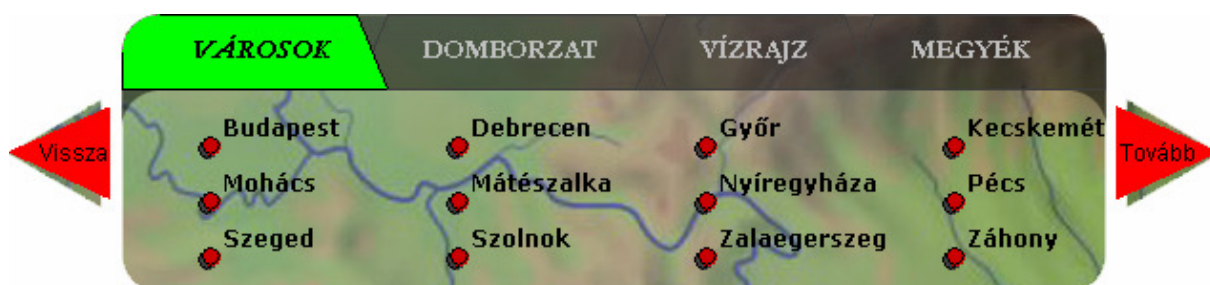


3.8.6. Névtár

Feladata és használata

Egy gyakorlat megoldása során Önnek földrajzi neveket kell elhelyeznie egy interaktív vaktérképen. Az oktatóalkalmazás vaktérképén jelenleg négyféle terepelem jeleníthető meg: települések, domborzati és vízrajzi elemek, illetve megyék. A *névtár* az adott topográfia gyakorlathoz tartozó tereptárgyaknak a földrajzi neveit tartalmazó felületi komponens, amely a képernyő jobb alsó sarkában található. A vaktérképen elhelyezkedő terepelemek nevei a gyakorlat megkezdését követően a névtárban helyezkednek el. A gyakorlat során innen kell a neveket átmozgatnia a térképen nekik megfelelő terepelem pozíciójába. A névtárban található nevek tehát az alkalmazás önálló, mozgatható elemei. A neveket a névtárban a vaktérképen ábrázolható terepelemeknek megfelelően négy kategóriából választhatja ki: településnevek, domborzati és vízrajzi elemek nevei, illetve megyenevek. A névtár minden egyes csoportjához tartozik egy-egy füles lap. Az egyes csoportokba tartozó földrajzi nevek az adott csoporthoz tartozó füles lapon jelennek meg. Ha egy adott csoportba sorolt földrajzi neveket szeretne elérni, akkor az adott csoporthoz tartozó füles lap fülére kell kattintania. Ennek hatására az Ön által látni kívánt nevek megjelennek a képernyőn. Amennyiben az adott csoportba tartozó földrajzi nevek nem férnek el a lap által elfoglalt képernyőterületen, akkor a jobbra-balra nyilak segítségével a lap tartalmát görgetheti a képernyőn.

Képernyőképek



Települések földrajzi nevei a névtárban

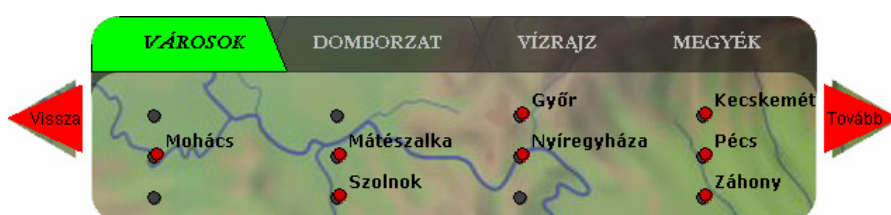
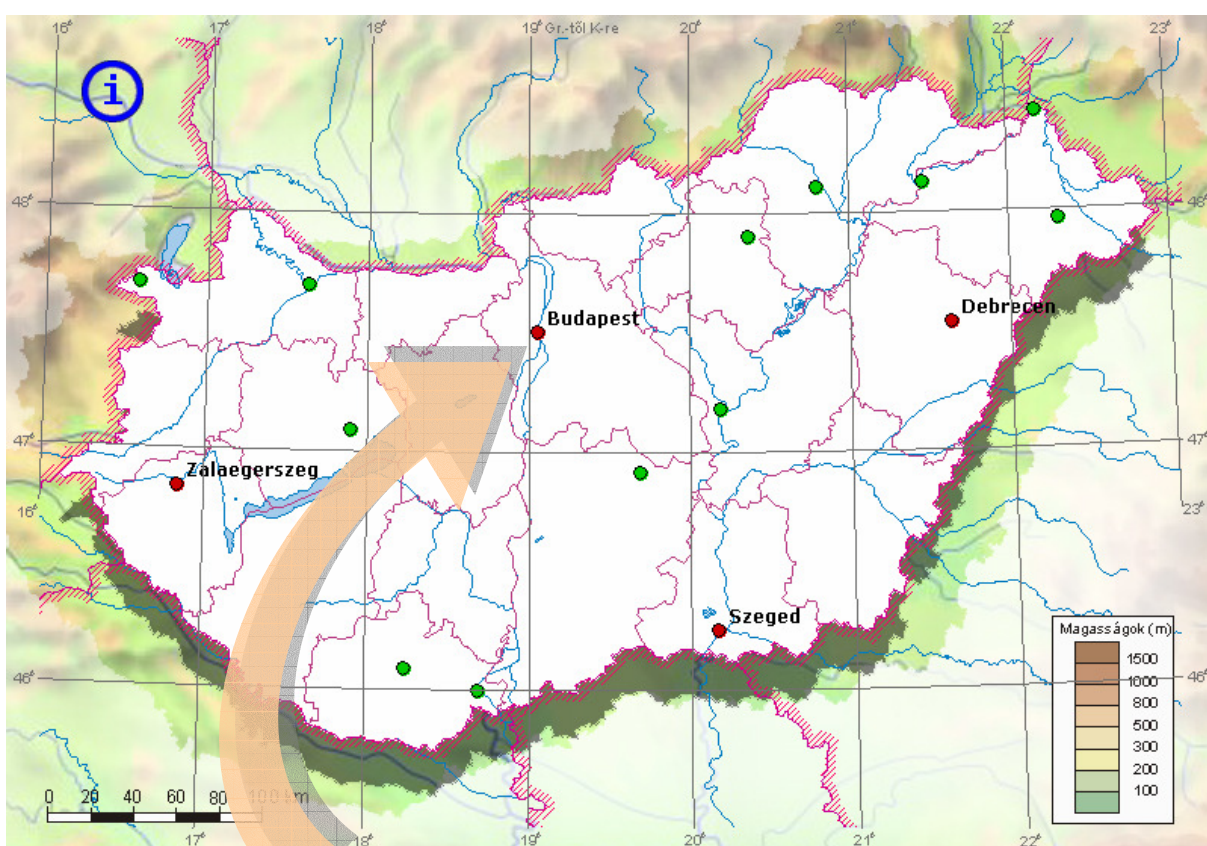


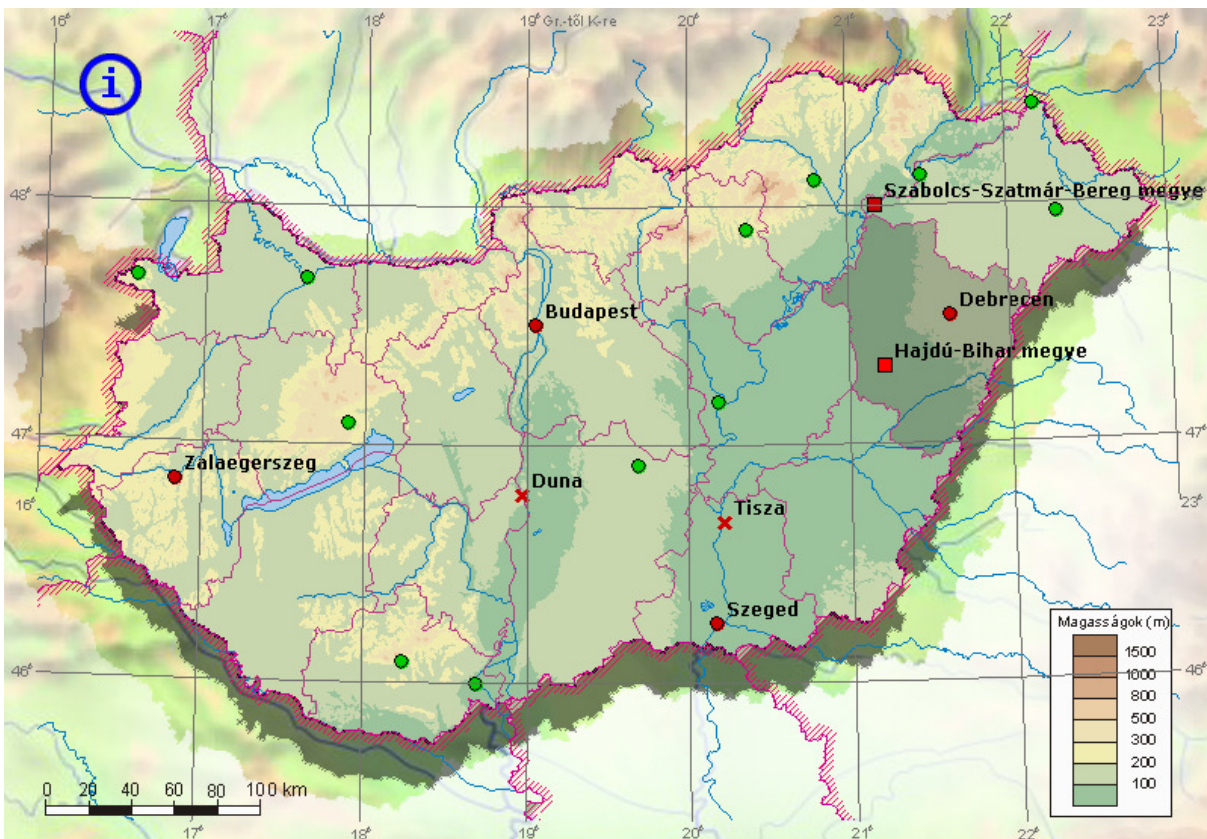
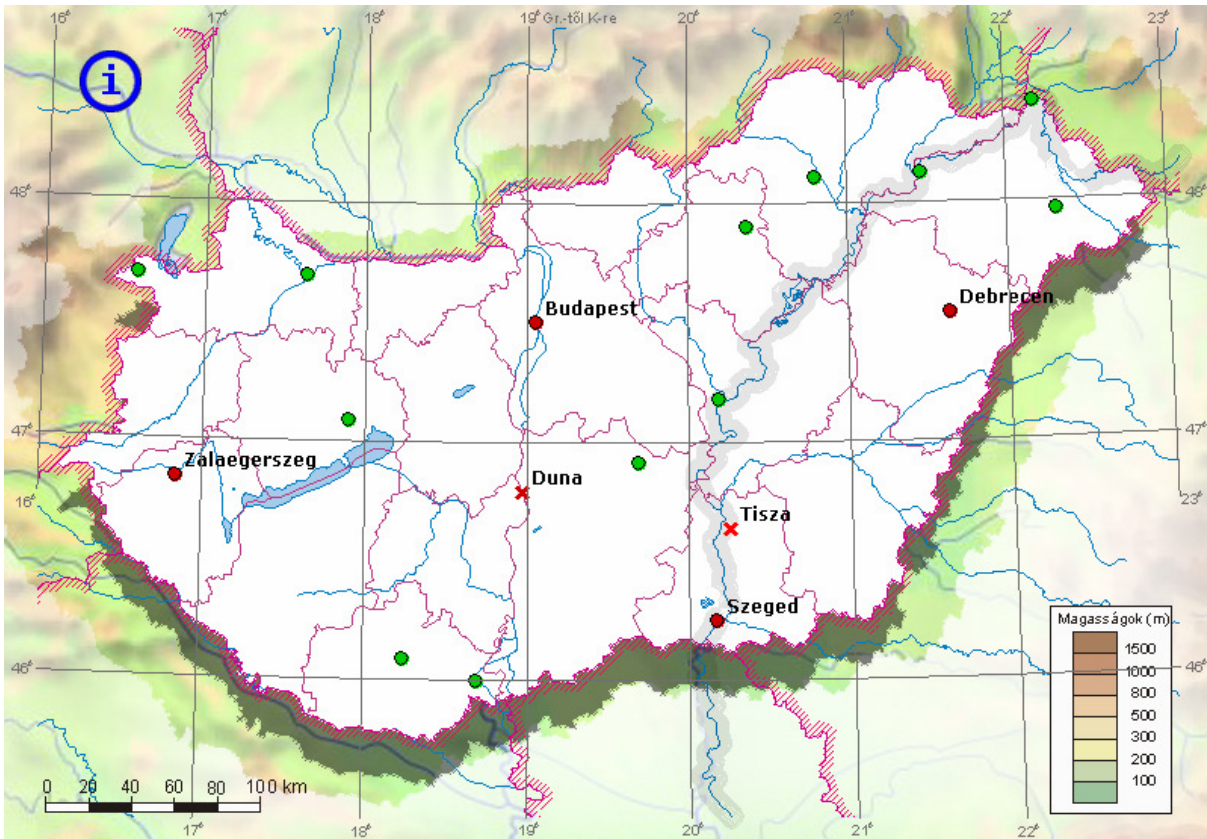
Megyék nevei a névtárban

3.8.7. Vaktérkép

Az alkalmazás legfontosabb vizuális komponense a *vaktérkép*. Egy topográfia gyakorlat megoldása során az Ön feladata, hogy a tanárai által az adott gyakorlatban oktatni, gyakoroltatni kívánt terepelemeket (települések, domborzati és vízrajzi elemek, megyék) és azok földrajzi neveit összepárosítsa, oly módon, hogy az adott terepelemhez tartozó földrajzi nevet kikeresi a névtárban, majd az egér segítségével átmozgatja azt a vaktérképen neki megfelelő terepelem pozíciójába. Ennek a műveletnek az elvégzéséhez az egér baloldali nyomógombjának lenyomásával meg kell ragadnia az adott nevet a névtárban, majd az egér bal gombjának nyomva tartása mellett a nevet át kell húznia a névtárból a neki megfelelő terepelemhez. Ha egy földrajzi név mozgatása közben egy terepelem fölé kerül, akkor az adott terepelem aktívvá válik, azaz sötétebb, illetve települések esetén világosabb színárnyalattal különül el a környezetétől.

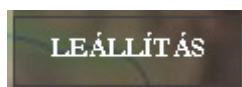
Képernyőképek





3.9. Gyakorlat befejezése

Az Ön által kiválasztott és elindított gyakorlat megoldását követően, az adott gyakorlat megoldásának befejezéséhez kattintson a gyakorló alkalmazás *Leállítás* feliratú nyomógombjára, amely az alkalmazás képernyőképének jobb felső sarkában található.

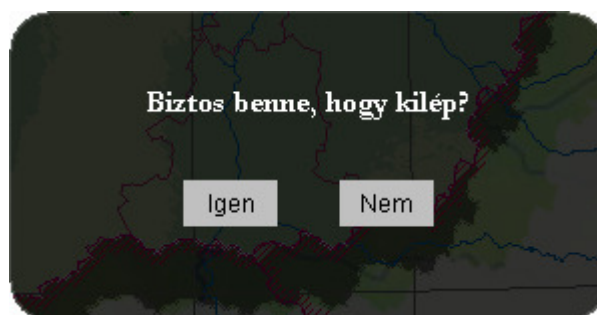


Ennek hatására az adott gyakorlat befejeződik, majd a program elvégzi a gyakorlat kiértékelését. Az értékelés eredményeként a helyesen pozícionált földrajzi nevek (települések, domborzati és vízrajzi elemek, illetve megyék nevei) fekete, a rosszul elhelyezett nevek pedig piros betűszínnel jelennek meg a térképen.

Ha egy gyakorlatot befejezett és az értékelést is megtekintette, lépjen ki a gyakorló alkalmazásból az alkalmazás képernyőképének jobb felső sarkában található *Kilépés* feliratú gomb segítségével.



A *Kilépés* gomb megnyomása után az alkalmazás megerősítést fog kérni a kilépési szándékáról:

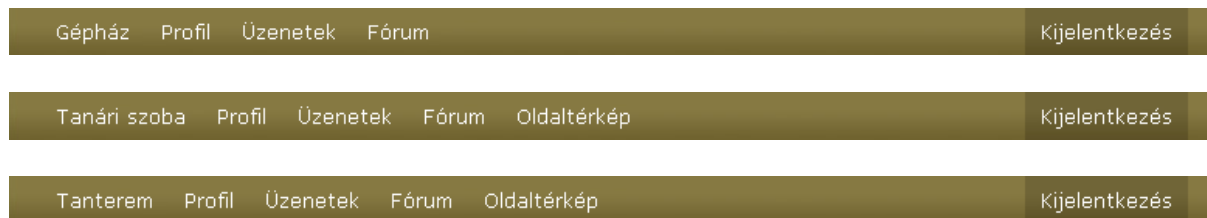


Ha szándéka szerint kattintott a *Kilépés* gombra, akkor az alkalmazásból történő kilépés folytatásához most válassza az *Igen* feliratú nyomógombot. Ennek hatására csak a gyakorló alkalmazás működése fejeződik be, a webes felületen keresztül azonban továbbra is lehetősége van újabb gyakorlatok kiválasztására és elindítására.

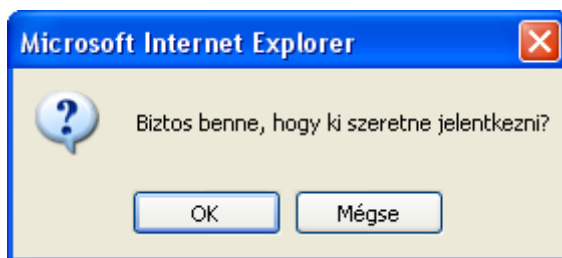
A *Nem* feliratú gomb megnyomásával visszatérhet a gyakorlathoz.

3.10. Kijelentkezés a rendszerből

Amennyiben minden gyakorlati teendőt elvégzett, jelentkezzen ki a rendszerből a menüsoron található *Kijelentkezés* menüpont kiválasztásával.



A menüpont kiválasztását követően az alkalmazás egy megerősítést fog kérni a kijelentkezési szándékáról:



Ha valóban szeretne kijelentkezni az alkalmazásból, akkor válassza az *Ok* feliratú nyomógombot. A *Mégse* feliratú gomb megnyomásával visszatérhet az alkalmazás normál működéséhez.

Sikeres kijelentkezést követően a topográfia oktatóalkalmazás nyitóoldalára kerül.

4. FEJLESZTŐI KÉZIKÖNYV

4.1. Bevezetés

Ez a fejezet a *Magyarország topográfija középiskolásoknak* webes oktatóalkalmazás fejlesztési útmutatója. A fejezet bemutatja a program technikai hátterét, általános felépítését, illetve a fejlesztés során alkalmazott irányelveket, megkötéseket.

A fejezet egyes alfejezetei gyakran tesznek utalásokat az alkalmazás osztályaira és objektumaira. Bár az említett osztályok, illetve objektumok szerepe és működése a legtöbb esetben ismertetésre kerül, az összefüggések pontosabb megértése érdekében mindenképpen érdemes az alkalmazás forráskódját is tanulmányozni.

4.2. Technikai háttér

Az alkalmazás Java és XML technológiákra épül. A következőkben felsorolom a program által használt fontosabb technológiákat.

Java technológiák:

- ▶ Java Applet technológia
- ▶ Java Servlet és JSP technológia
- ▶ EJB technológia

XML technológiák:

- ▶ XSL átalakító (XSLT)
- ▶ Extensible Hypertext Markup Language (XHTML)
- ▶ XML Path Language (XPath)


Ezen technológiák ismerete a program szerkezetének és működésének megértéséhez elengedhetetlen, ismertetésükre azonban ebben a dolgozatban területi és tartalmi okok

miatt nem kerülhetett sor. Java Appletekről [1]-ben, a Java Servlet technológiáról [1]-ben és [2]-ben, az EJB technológiáról [2]-ben, magáról az XML leíró nyelvről [2]-ben és [4]-ben, a fent felsorolt XML technológiákról pedig [4]-ben olvashat részletesebben.

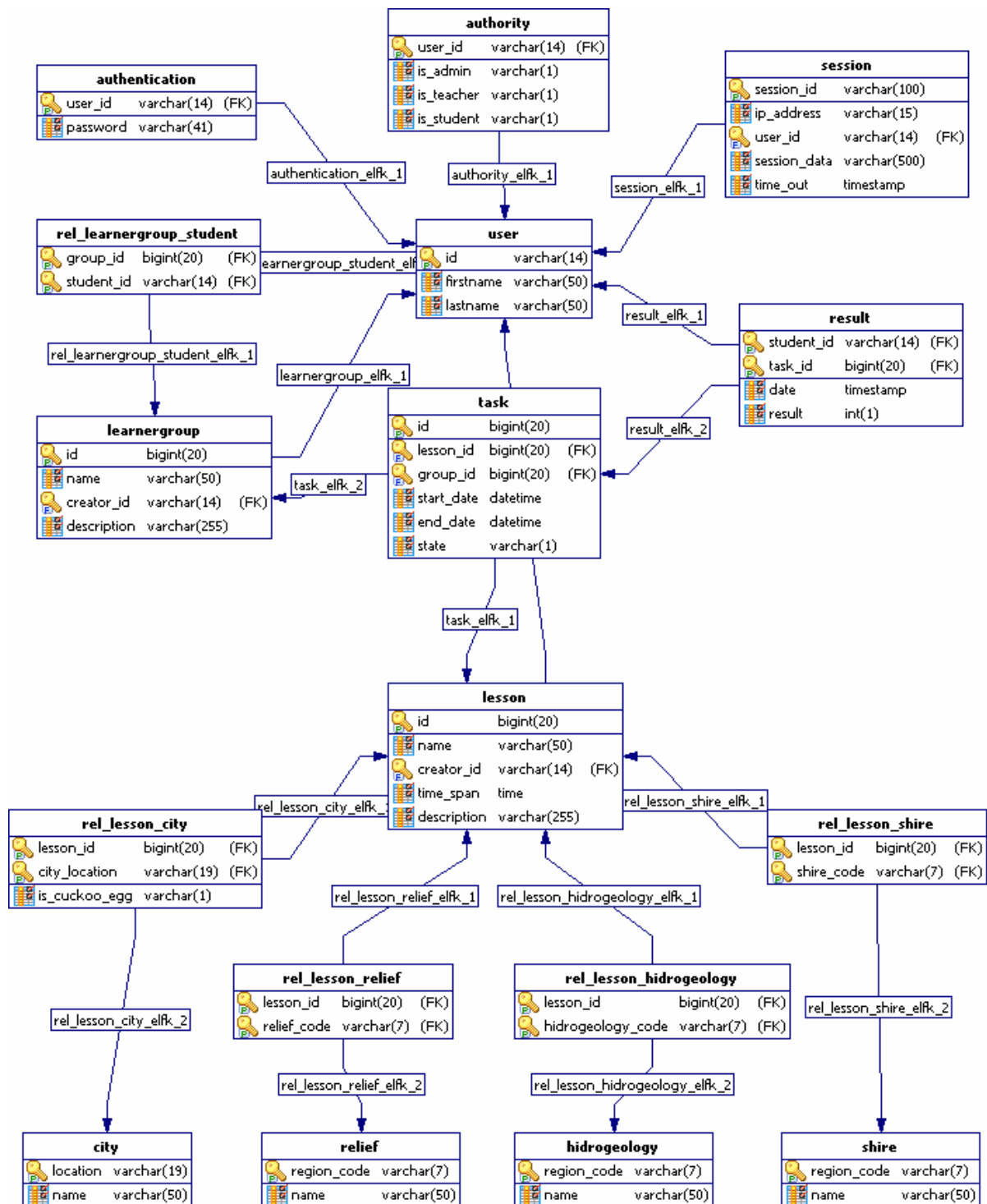
4.3. Fejlesztői környezet

Az alkalmazás elkészítéséhez az ingyenesen használható, 3.2-es verziójú Eclipse integrált szoftverfejlesztői környezetet használtam. Az Eclipse egy platformfüggetlen, nyílt forráskódú (open source) alkalmazás, amely széleskörű támogatást nyújt Java alkalmazások készítéséhez, és az interneten ingyenesen elérhető több száz bővítmény segítségével szinte az összes ma használatos fejlesztési technológiát támogatja. (Lásd még Függelék 7.1 fejezet!)

4.4. Az alkalmazás adatbázisának felépítése

Az alkalmazás működése közben felhasznált és kezelt adatok az *elearn* nevű adatbázisban tárolódnak. A program által használt adatbázis adattábláit és azok kapcsolatát az *1. ábra* szemlélteti. Az ábrán *rel_* előtaggal kezdődő táblanevek kapcsoló táblák nevei, amelyeket két másik tábla N:M típusú összekapcsolásához használok, és amelyeknek sorai azonosító adatokat rendelnek egymáshoz. Például a *rel_learnergroup_student* tábla minden sora egy tanulócsoporthoz azonosítóhoz (*group_id*) egy tanuló jogosultsági csoportba tartozó felhasználói azonosítót (*student_id*) rendel. Az ábrán  ikonnal megjelölt mezők kulcsok. Az egyes táblák elsődleges kulcsainak neveit vízszintes vonallal választottam el a többi mező nevéétől. A táblák az elsődleges kulcsok szerint indexelve vannak, így az adott mezőnevek szerinti adatkeresés gyorsabb. Az idegen kulcsokat a mezők neve mellett (FK)-val jelöltem, a megszorítások neveit pedig a kapcsolatokat jelölő nyilakon adtam meg. Az egyes mezőnevek mellett feltüntettem a mezők típusát, illetve ahol lehetséges, ott az adott típusnév mellett zárójelben az ábrázolás hosszát is megadtam.

A következőkben ismeretem az adatbázis egyes tábláinak szerepét:



1. ábra Az adatbázis tábláinak kapcsolata

4.4.1. Az authentication tábla

Az alkalmazás ebben a táblában tárolja a felhasználók azonosító adatait (felhasználói azonosító, jelszó). Bejelentkezéskor a program a felhasználók azonosítását ezen tábla adatai alapján végzi el. A táblában a jelszavak az adatbázis-kezelő rendszer `PASSWORD(str)` függvényével vannak titkosítva.

4.4.2. Az authority tábla

Ebben a táblában tárolódnak a felhasználók jogosultságai. Az alkalmazás a felhasználói csoportoknak megfelelően három jogosultságot különböztet meg: az adminisztrátori, a tanári és a tanulói jogosultságot. Minden felhasználó pontosan egy felhasználói csoportba tartozhat, és pontosan egy jogosultsággal rendelkezhet. Az adott jogosultság meglétét a mezők **'I'**, hiányát pedig **'N'** értéke jelzi.

A következő táblázat az alkalmazással együtt telepíthető tesztadatok esetében szemlélteti az egyes felhasználók jogosultságait.

user_id	is_admin	is_teacher	is_student
admin	I	N	N
student1	N	N	I
student2	N	N	I
student3	N	N	I
teacher1	N	I	N
teacher2	N	I	N

4.4.3. A session tábla

Ebben a táblában a program a bejelentkezett felhasználók kapcsolatára vonatkozó információkat tárol. Ilyen információ pl. a felhasználó számítógépének IP címe, vagy a felhasználó böngészőprogramjának típusa, verziószáma. Ezek az adatok ideiglenesen, a felhasználó kijelentkezéséig tárolódnak.

4.4.4. A user tábla

Ebben a táblában tárolja a program a regisztrált felhasználók egyéb személyes adatait (jelenleg csak a felhasználók azonosítóit és nevét).

4.4.5. A learnergroup tábla

Ebben a táblában tárolja a program a tanulócsoportokkal kapcsolatos adatokat, a tanulócsoportok nevét, rövid leírását, a csoportot létrehozó felhasználó azonosítóját és a csoportazonosítót. A csoporthoz tartozó tanulók azonosítóit és a csoportazonosítót a *rel_learnergroup_student* kapcsolótábla rendeli egymáshoz.

4.4.6. A city tábla

Ebben a táblában tárolódnak a gyakorlatok során számonkérhető és az interaktív vaktérképen elhelyezhető települések földrajzi koordinátái és nevei. A települések földrajzi koordinátái egyetlen mezőben (*location*) [*északi szélesség, keleti hosszúság*] formátumban vannak letárolva, ahol a szélességi és hosszúsági koordináták *fok:perc.másodperc* formában, 8 karakteren rögzített értékek. Ezen jelölésmód mellett Debrecen földrajzi koordinátái pl. a következőképpen adhatóak meg: [*47:31.73,21:38.23*].

4.4.7. A hidrogeology tábla

Ebben a táblában tárolódnak a topográfia gyakorlatok során számonkérhető vízrajzi elemek azonosító kódjai (*region_code*) és nevei. Az azonosító kód egy *#hhhhhh* formátumban megadott RGB színkód, ahol *h* egy 16-os számrendszerbeli hexadecimális számjegyet jelöl. A színkódot az alkalmazás az adott terepelemnek a következő oldalon látható speciális térképen történő beazonosításához használja. Ez a térkép minden vízrajzi elemet egyetlen, de elemenként különböző színárnyalattal ábrázol. Az alkalmazás ezzel a módszerrel határozza meg az elemek elhelyezkedését és kiterjedését. A domborzati elemek és megyék helyzetének meghatározásához az alkalmazás ugyanezt az eljárást használja.



4.4.8. A relief tábla

Ebben a táblában tárolódnak a domborzati elemek azonosító kódjai (*region_code*) és nevei. Az azonosító kód egy *#hhhhh* formátumban megadott RGB színrendszerbeli színkód. (A színkód értelmezését és szerepét lásd a 4.2.7 fejezetben!)

4.4.9. A shire tábla

Ebben a táblában tárolódnak az egyes megyék azonosító kódjai (*region_code*) és nevei. Az azonosító kód egy *#hhhhh* formátumban megadott RGB színkód. (A színkód értelmezését és szerepét lásd a 4.2.7 fejezetben!)

4.4.10. A lesson tábla

A program ebben a táblában tárolja a leckék adatait, azok nevét, rövid leírását, a lecke megoldására javasolt időintervallum hosszát, a leckét készítő felhasználó azonosítóját, illetve a leckeazonosítót. A lecke készítője (*creator_id*) egy tanár jogosultsági csoportba tartozó felhasználó lehet. A lecke megoldása során gyakoroltatni kívánt tereptárgyak azonosítóit és a

leckeazonosítót a *rel_lesson_city*, *rel_lesson_relief*, *rel_lesson_hidrogeology*, *rel_lesson_shire* kapcsolótáblák rendelik egymáshoz.

4.4.11. A task tábla

Az alkalmazás ebben a táblában tárolja a gyakorlatokkal kapcsolatos információkat, a gyakorlat során oktatni kívánt lecke egyedi azonosítóját, illetve annak a tanulócsoporthoz az azonosítóját, amely számára az adott gyakorlat megoldását egy tanár jogosultságú felhasználó, a gyakorlat létrehozója előírta, továbbá a gyakorlat ütemtervének dátumait és a gyakorlat saját egyedi azonosítóját.

4.4.12. A result tábla

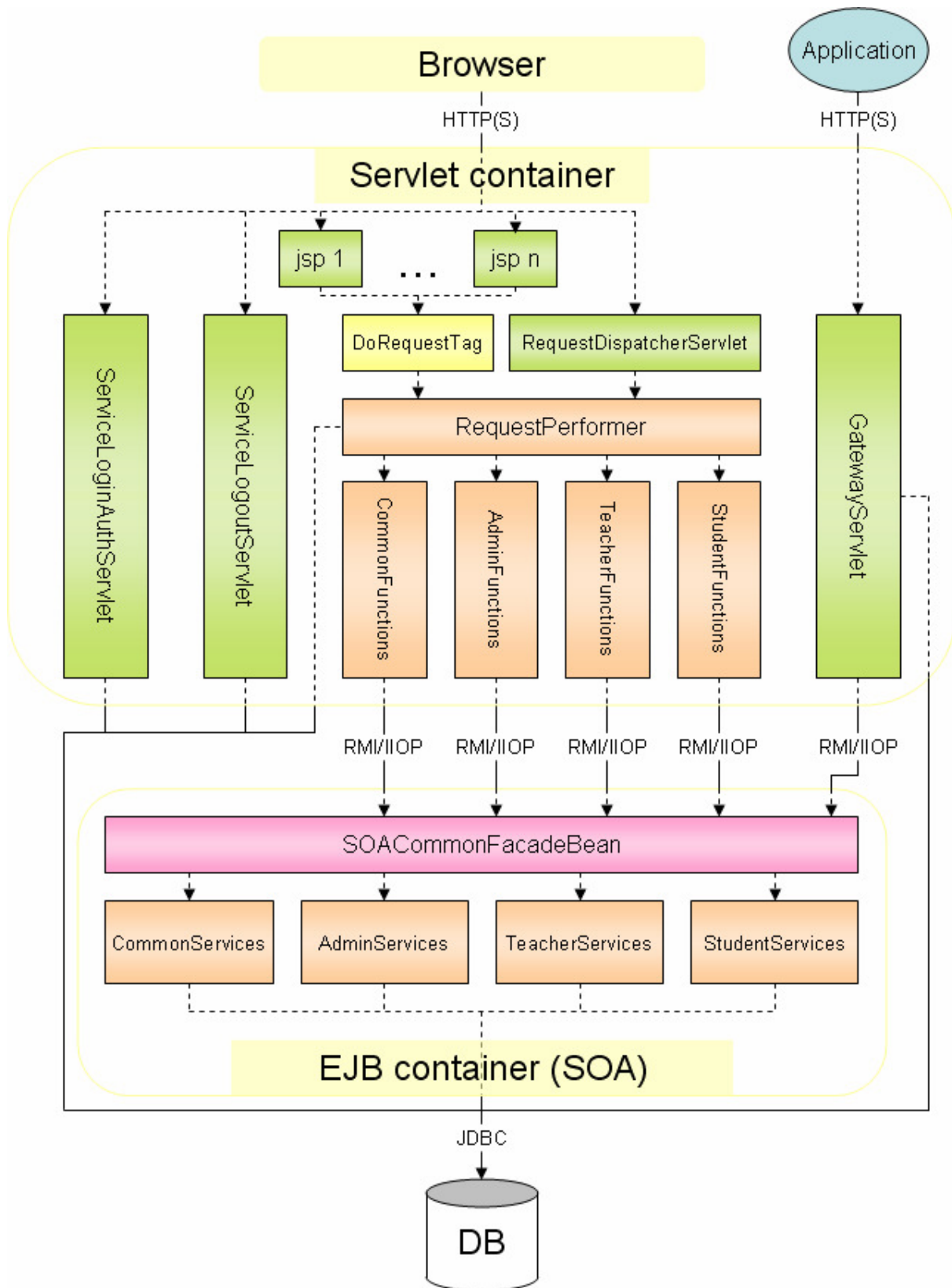
Ebben a táblában tárolódnak az egyes tanulók által a gyakorlat legutóbbi megoldása során elért pontszámok és az utolsó megoldás dátuma.

4.5. Az alkalmazás szerkezete

Amint azt a 3.3 fejezetben már olvashatta az alkalmazás két, egymással szorosan együttműködő egységből épül fel, a keretalkalmazásból és a topográfiai alkalmazásból. A továbbiakban ezek szerkezetét és működését ismertetem.

4.6. A keretalkalmazás felépítése, működése

A keretalkalmazás szerkezeti diagramját a *2. ábra* szemlélteti. Az ábrán látható téglalapok az alkalmazás fontosabb osztályait és objektumait jelölik, a szaggatott vonalak a kommunikáció irányát, míg a folytonos vonalak a fizikai csatornákat és azok típusát szemléltetik. Az ábrán kék (■) háttérszínű ellipszissel a topográfiai alkalmazást is feltüntettem.



2. ábra Az alkalmazás szerkezeti diagramja

A 2. ábrán barna (■) háttérszínnel megadott téglalapok egyszerű Java osztályok*, a rózsaszín (■) téglalap EJB objektumot jelöl, a zöld (■) színűek Java Servletek és JSP oldalak, a sárga (■) háttérszínű téglalap pedig egy saját JSP akció-elem objektuma.

Az alábbiakban ismeretem az egyes osztályokat és objektumokat:

4.6.1. JSP oldalak (■)

Az alkalmazás weboldalait előállító JSP állományok a *ClientAdminWAR* projekt *WebContent/pages* mappájában találhatóak. A következő táblázatban felsorolom az alkalmazás weboldalait, és minden oldal mellett megadom az adott oldalt előállító JSP állományok nevét.

Weboldal neve	JSP állomány neve
Bejelentkező oldal	login.jsp
Gépház	adminservlets.jsp
Tanári szoba	teacherservlets.jsp
Tanári szoba – Lecke létrehozása	createlesson.jsp
Tanári szoba – Gyakorlat létrehozása	createtask.jsp
Tanterem	studentservlets.jsp
Gyakorlat oldal	client.jsp
Hibaoldal	error.jsp

A JSP oldalak a kimenetet XHTML formátumban generálják le. Az XHTML leíró nyelvről [4]-ben olvashat részletesebben.

A következőkben megadom az alkalmazás fejlesztése során elkészített saját JSP akcióelemek listáját. Az akcióelemek és azok legfontosabb tulajdonságai táblázatos formában láthatóak. Az elemek implementációi a *CommonTagLib* és a *ClientAdminTagLib* projektekben találhatóak. (Lásd 7.1 fejezet!)

* Olyan osztályok, amelyekből egy ClassLoader alatt egyetlen példány sem létezik. Az egyszerű Java osztályokhoz való hozzáférést az osztályokban definiált osztályszintű metódusok biztosítják

akcióelem	
név	setHeader
osztály	com.dns.common.jsp.handler.SetHeaderTag
projekt	CommonTagLib
attribútum	
név	name
típus	java.lang.String
kötelező	igen
attribútum	
név	value
típus	java.lang.String
kötelező	igen

A HTTP protokollt használó kommunikációban az üzenetek (kérés- és válaszüzenetek egyaránt) úgynevezett fejlécmezőket (headers) tartalmazhatnak, amelyek különböző információt hordoznak a keresőrobotok és a böngészőprogramok számára az üzenet tartalmáról, annak kezeléséről, tárolásáról, stb. Bizonyos szabályok betartása mellett mi magunk is elhelyezhetünk adatokat ezekben a fejlécmezőkben, illetve arra is van lehetőség, hogy a HTTP üzenet fejlécébe új mezőket vegyünk fel. Egy mező formálisan a következőképpen adható meg: *név=érték*. A `<setHeader name="str" value="str"/>` akcióelem a HTTP válasz fejlécének beállításához használható. Az akcióelem működése: Ha az adott nevű mező létezik a HTTP válasz fejlécében, akkor annak értékét felülírja a *value* argumentumban megadott értékkel. Ha a név nem szerepel egyetlen fejlécmezőben sem, akkor az elem egy új mezőt, azaz egy név-érték párost ad a válasz fejlécéhez.

akcióelem	
név	redirect
osztály	com.dns.common.jsp.handler.RedirectTag
projekt	CommonTagLib
attribútum	
név	url
típus	java.lang.String
kötelező	igen

A `<redirect url="str"/>` akcióelem a böngészőprogram adott oldalra történő átirányításához használható. Az *url* attribútum értékeként abszolút és relatív cím egyaránt megadható. A relatív hivatkozásokat az akcióelem az őt használó JSP oldalhoz viszonyítva értelmezi.

akcióelem	
név	obfuscate
osztály	com.dns.common.jsp.handler.ObfuscateTag
projekt	CommonTagLib
attribútum	
név	level
típus	int
kötelező	nem

A `<obfuscate level="integer">str</obfuscate>` akcióelem a JSP oldal által generált XHTML formátumú szöveg „homályosításához” használható. A „homályosítás” erősségét a *level* argumentumban adhatjuk meg. Ha nem adunk meg argumentumot, akkor a szöveg nem lesz „elhomályosítva”.

akcióelem	
név	insertCurrentDate
osztály	com.dns.common.jsp.handler.InsertCurrentTag
projekt	CommonTagLib
attribútum	
név	format
típus	java.lang.String
kötelező	igen

Az `<insertCurrentDate format="str" />` akcióelem az aktuális dátum beillesztéséhez használható. A dátumformátum a *format* argumentumban adható meg.

akcióelem	
név	ifLoggedIn
osztály	com.dns.client.admin.jsp.handler.IfLoggedInTag
projekt	ClientAdminTagLib

Az `<ifLoggedIn>str</ifLoggedIn>` akcióelem nyitó- és záróelemei között megadott szöveg csak bejelentkezett felhasználók esetén kerül bele a kimenetbe.

akcióelem	
név	ifNotLoggedIn
osztály	com.dns.client.admin.jsp.handler.IfNotLoggedInTag
projekt	ClientAdminTagLib

Az `<ifNotLoggedIn>str</ifNotLoggedIn>` akcióelem nyitó- és záróelemei között megadott szöveg csak nem bejelentkezett felhasználók esetén kerül bele a kimenetbe.

akcióelem	
név	doRequest
osztály	com.dns.client.admin.jsp.handler.DoRequestTag
projekt	ClientAdminTagLib
attribútum	
név	requestId
típus	java.lang.String
kötelező	igen
attribútum	
név	templateName
típus	java.lang.String
kötelező	nem

A `<doRequest requestId="str" templateName="str"/>` akcióelem segítségével a JSP oldalak elérhetik az alkalmazás funkcióit. Az adott funkciót a *requestId* argumentum értéke határozza meg, míg a *templateName* argumentumban azt lehet megadni, hogy a válasz objektum melyik XSL sablon szerint kerüljön feldolgozásra. Az alkalmazás funkcióiról a 4.6.7-4.6.10 fejezetekben, a válaszobjektumok feldolgozásáról pedig a 4.6.4-4.6.6 fejezetekben olvashat részletesebben.

akcióelem	
név	ifWasError
osztály	com.dns.client.admin.jsp.handler.IfWasErrorTag
projekt	ClientAdminTagLib
attribútum	
név	errorCode
típus	int
kötelező	nem

Az `<ifWasError errorCode="integer">str</ifWasError>` akcióelem nyitó- és záróelemei között megadott szöveg csak akkor kerül bele kimenetbe, ha az *errorCode* argumentumban megadott azonosítóval rendelkező hiba bekövetkezett. Ha az *errorCode* argumentumot elhagyjuk, akkor a nyitó- és záróelemek között elhelyezett szöveg bármely hiba esetén bekerül a kimenetbe.

4.6.2. A ServiceLoginAuthServlet osztály (■)

Az osztály a *ClientAdminWAR* projektben, a *com.dns.client.admin* csomagban található. Amikor a felhasználó a *Bejelentkező oldalon (login.jsp)* kitölti a *Bejelentkező űrlapot* (Lásd 3.6.2 fejezet!), és az egérrel a *Bejelentkezés* gombra kattint, akkor a böngészőprogram az űrlap adatait egy HTTP kérés törzsében elküldi az alkalmazáservernek. Az alkalmazáserveren a kérést a *ServiceLoginAuthServlet* osztály egy objektuma dolgozza fel. A szervletobjektum az elküldött adatok alapján ellenőrzi a felhasználó jogosultságát, majd sikeres azonosítást követően létrehozza a munkamenet objektumot* (*javax.servlet.http.HttpSession*), eltárolja a felhasználó kapcsolatára jellemző adatokat a *session* adatbázistáblában, végül a böngészőt átirányítja a felhasználó jogosultságának megfelelő oldalra (*Gépház/Tanári szoba/Tanterem*).

4.6.3. A ServiceLogoutServlet osztály (■)

Az osztály a *ClientAdminWAR* projektben, a *com.dns.client.admin* csomagban található. Miután egy felhasználó a felső menüsoron a *Kijelentkezés* menüpontra kattint, és a megjelenő párbeszédablakban megerősíti a kijelentkezési szándékát, a böngészőprogram egy HTTP kérést küld az alkalmazáservernek. A kérés hatására a *ServiceLogoutServlet* osztály egy példánya megszünteti a felhasználóhoz rendelt munkamenetet, és törli a kapcsolat információit a *session* adattáblából, majd a böngészőt átirányítja a *Bejelentkező oldalra*.

4.6.4. A RequestDispatcherServlet osztály (■)

A bejelentkezett felhasználók által kitöltött és elküldött űrlapokon megadott adatokat a böngészőprogram HTTP kérések törzsében küldi el az alkalmazáservernek. A serveren a kérést a *RequestDispatcherServlet* osztály egy objektuma dolgozza fel. A szervletobjektum a kérés feldolgozásához meghívja a *RequestPerformer* osztály

* Egy felhasználó és a kiszolgáló között “fennálló” kapcsolatot jelképez. Az alkalmazáserver (*a háttérben pl. cookie-k segítségével*) azonosítja a felhasználót és a böngészőt, így a felhasználó a böngésző bezárásáig, illetve a munkamenet objektum érvénytelenítéséig (kijelentkezés) saját munkamenetébe térhet vissza.

```

public static java.lang.Object process(HttpServletRequest request)
    throws AuthorizationFailureException, SessionHandlingException,
        RequestNotFoundException, RequestFormatException;

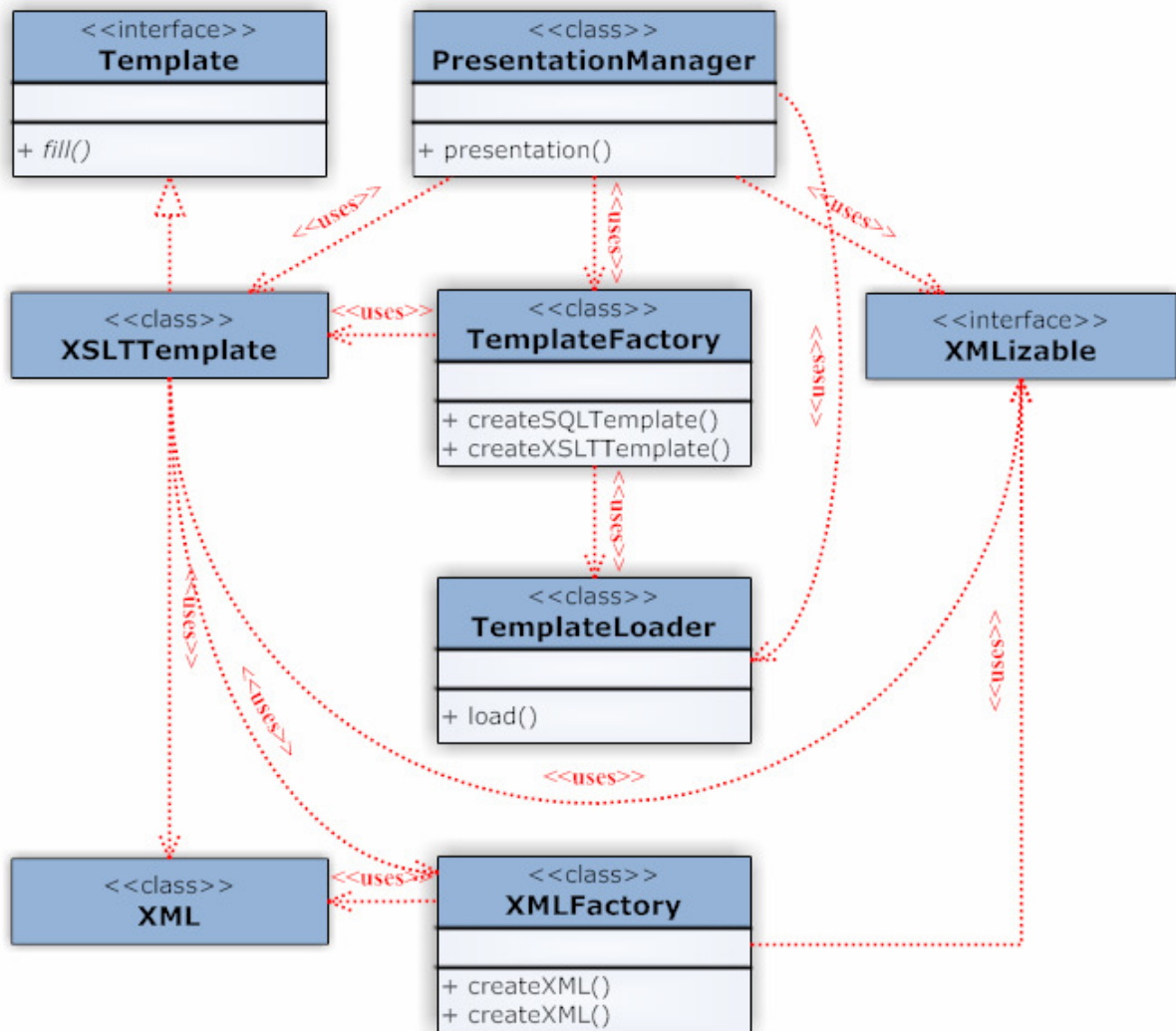
```

metódusát paraméterben átadva neki a kérésobjektumot. A kérés feldolgozását a *RequestPerformer* osztály vezérli le. A feldolgozást követően a szervletobjektum a böngészőprogramot a kéréstől függően átirányítja az alkalmazás megfelelő weboldalára. Azt, hogy a böngésző a kérés kiszolgálása után melyik weboldalt fogja letölteni, a kérés azonosítója határozza meg. Az azonosítót a *RequestDispatcherServlet* példány a kérésobjektumban paraméterként kapja meg (*request_id*).

4.6.5. A DoRequestTag osztály

Ha egy JSP oldalnak a kimenet előállításához olyan adatokra van szüksége, amelyek helyben nem állnak rendelkezésre, pl. egy tanár leckéit le kell kérdezni az adatbázisból, akkor a JSP oldalnak meg kell hívnia az alkalmazás egy funkcióját, amely a szükséges adatokat előállítja számára. Egy JSP oldal a `<doRequest requestId="str" templateName="str" />` saját akcióelem segítségével érhet el egy funkciót. A *DoRequestTag* osztály a *doRequest* nevű akcióelem elemkezelő (tag handler) Java osztálya, a *javax.servlet.jsp.tagext.Tag* osztály leszármazottja. A kívánt funkció eléréséhez a *DoRequestTag* elemkezelő osztály egy példánya a *RequestPerformer* osztály *process* metódusát használja. A jogosultságok ellenőrzését, és a megfelelő funkció meghívását a *RequestPerformer* osztály *process* metódusa végzi. A *DoRequestTag* osztály a kért adatokat a *process* metódus visszatérési értékeként egy XML formátumú szöveggé alakítható, *com.dns.common.base.xml.XMLizable* osztályú válasz objektumba csomagolva kapja meg. A válasz objektum feldolgozását a *PresentationManager* osztály végzi el. A feldolgozás a *doRequest* akcióelem *templateName* attribútumában megadott, vagy annak hiányában a *requestId*-val megegyező nevű, és XSL formátumú stíluslapot tartalmazó sablon állomány segítségével történik. A feldolgozás eredményeként előálló XHTML formátumú szöveget a *DoRequestTag* osztály beilleszti a JSP oldal kimenetébe a *doRequest* akcióelem helyére.

A *PresentationManager* osztály és a válasz objektum feldolgozása során általa felhasznált objektumok osztályainak UML diagramját a 3. ábra szemlélteti.



3. ábra A válasz objektumból XHTML formátumú szöveget generáló osztályok diagramja

A válasz objektumok feldolgozására és a *PresentationManager* osztály működésére példaként tekintünk a következő oldalon látható két osztályt. A példában bemutatott *com.dns.soa.types.common.GetUsersInfoResponse* osztály egy konkrét objektuma létrejöhet például a *getLearnerGroupStudents* funkció meghívását követően, annak válasz objektumaként. A *getLearnerGroupStudents* funkció szerepéről a 4.6.7 fejezet ír bővebben. Figyeljük meg, hogy mindkét osztály implementálja az *XMLizable* interfészt.

```

1. public class GetUsersInfoResponse implements XMLizable, Serializable {
2.     private static final long serialVersionUID = 4838347131288454498L;
3.
4.     private GetUsersInfoResponseItem[] items;
5.
6.     public void setItems(GetUsersInfoResponseItem[] items) {
7.         this.items = items;
8.     }
9.
10.    public GetUsersInfoResponseItem[] getItems() {
11.        return items;
12.    }
13. }

```

```

1. public class GetUsersInfoResponseItem implements XMLizable,
2.     Serializable {
3.     private static final long serialVersionUID = 3294285871817323578L;
4.
5.     private String id;
6.     private String firstName;
7.     private String lastName;
8.
9.     public void setId(String id) {
10.        this.id = id;
11.    }
12.
13.    public String getId() {
14.        return id;
15.    }
16.
17.    public void setFirstName(String firstName) {
18.        this.firstName = firstName;
19.    }
20.
21.    public String getFirstName() {
22.        return firstName;
23.    }
24.
25.    public void setLastName(String lastName) {
26.        this.lastName = lastName;
27.    }
28.
29.    public String getLastName() {
30.        return lastName;
31.    }
32. }

```

A *GetUsersInfoResponse* osztály egy konkrét állapottal rendelkező objektumából, vagyis egy konkrét válaszbjektumból a [3. ábrán](#) látható *XMLFactory* osztály *createXML* nevű statikus metódusa XML formátumú szöveget generál. Egy lehetséges példa az *XMLFactory* osztály által generált XML-re:

```

1. <?xml version="1.0" encoding="ISO-8859-2"?>
2.
3. <GetUsersInfoResponse>
4.     <GetUsersInfoResponseItem>
5.         <id>gipszj</id>
6.         <firstName>Jakab</firstName>
7.         <lastName>Gipsz</lastName>
8.     </GetUsersInfoResponseItem>
9.     <GetUsersInfoResponseItem>
10.        <id>gipsze</id>
11.        <firstName>Elek</firstName>
12.        <lastName>Gipsz</lastName>
13.    </GetUsersInfoResponseItem>
14. </GetUsersInfoResponse>

```

Ezek után tekintsük a következő XSL formátumban megadott szöveget, amellyel a 3. ábrán látható *com.dns.common.base.template.xslt.XSLTemplate* osztály a fenti XML formátumú szöveget XHTML formátumú szöveggé alakítja át:

```

1. <?xml version="1.0" encoding="ISO-8859-2"?>
2.
3. <xsl:stylesheet version="1.0"
4. xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5.     <xsl:template match="/">
6.         <SELECT name="users">
7.             <OPTION></OPTION>
8.             <xsl:for-each select="GetUsersInfoResponse">
9.                 <xsl:apply-templates/>
10.            </xsl:for-each>
11.        </SELECT>
12.    </xsl:template>
13.
14.    <xsl:template match="GetUsersInfoResponseItem">
15.        <OPTION>
16.            <xsl:value-of select="lastName"/>
17.            <xsl:value-of select="firstName"/>
18.        </OPTION>
19.    </xsl:template>
20. </xsl:stylesheet>

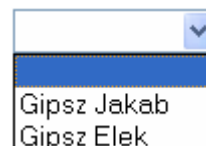
```

A feldolgozást követően eredményül kapott XHTML formátumú szöveg:

```

1. <SELECT name="users">
2.     <OPTION></OPTION>
3.     <OPTION>
4.         Gipsz Jakab
5.     </OPTION>
6.     <OPTION>
7.         Gipsz Elek
8.     </OPTION>
9. </SELECT>

```



4. ábra

A baloldali XHTML kód által leírt komponens

4.6.6. A RequestPerformer osztály (■)

A *RequestPerformer* osztály a keretalkalmazás űrlapelemeiről érkező adatok feldolgozását, illetve a JSP oldalak *doRequest* akcióelemeivel elküldött kérések feldolgozását végzi. Az osztálynak egyetlen statikus metódusa van:

```
public static java.lang.Object process(HttpServletRequest request)
    throws AuthorizationFailureException, SessionHandlingException,
    RequestNotFoundException, RequestFormatException;
```

A metódus által dobott kivételeket a hívó objektum (a *RequestDispatcherServlet* vagy a *DoRequestTag* osztály egy objektuma) kezeli. A kérés feldolgozása előtt a metódus ellenőrzi, hogy van-e jogosultsága a felhasználónak a kérés végrehajtásához. Amennyiben a felhasználó nem rendelkezik a megfelelő jogosultságokkal, akkor a metódus eldob egy *AuthorizationFailureException* kivételobjektumot. Ha a hívó *RequestDispatcherServlet* objektum kivételkezelője elkap egy *AuthozitaionFailureException* osztályú kivételt, akkor a böngészőt átirányítja az alkalmazás *Bejelentkező oldalára*. A *doRequest* akcióelemem használatakor ez a kivétel nem következhet be. Ha a jogosultságellenőrzés sikeres volt, akkor a feldolgozó *RequestPerformer* osztály frissíti a *session* táblában tárolt adatokat, majd a kérésazonosító alapján meghívja a megfelelő funkciót. A kérés feldolgozását a továbbiakban az adott funkció végzi. A végrehajtható funkciók kódját a *CommonFunctions*, *AdminFunctions*, *TeacherFunctions* és *StudentFunctions* osztályok implementálják. Ha a *session* tábla adatainak frissítése – például adatbázis hiba miatt – meghiúsul, akkor erről a metódus egy *SessionHandlingException* kivételobjektum dobásával tájékoztatja a hívót. Ha a meghívni kívánt funkció nincs implementálva, akkor a metódus *RequestNotFoundException* kivételt dob, *RequestFormatException* kivétel eldobásával pedig azt jelzi a program, hogy a kivétel a funkció végrehajtása során a felhasználó hibájából, például hibás adatbevitel miatt történt. Ebben az esetben a hívó csak egy *RequestDispatcherServlet* objektum lehet, és a hívó objektum a hibakezelést arra a JSP oldalra bízta, amelyen az űrlapot kitöltötték.

4.6.7. A CommonFunctions osztály (■)

Ebben az osztályban vannak implementálva azok a funkciók, amelyeket az alkalmazás tetszőleges jogosultsági csoportjába tartozó felhasználója meghívhat. Egy funkció az osztály

egy statikus metódusának meghívásával érhető el. A funkciók meghívását a *RequestPerformer* osztály *process* metódusa végzi. Azt, hogy egy adott kérés során melyik funkció dolgozhatja fel a kérést a kérésazonosító alapján dönti el az alkalmazás. A kérésazonosítók és a metódusok megfeleltetését a következő táblázat tartalmazza:

Kérésazonosító	Metódus neve	A válaszbjektum osztálya
<i>getUserName</i>	<i>getUserName</i>	<i>GetUsersInfoReponse</i>
<i>getLearnerGroupStudents</i>	<i>getLearnerGroupStudents</i>	<i>GetUsersInfoReponse</i>
<i>getAllCityNames</i>	<i>getAllCityNames</i>	<i>GetCitiesResponse</i>
<i>getAllReliefNames</i>	<i>getAllReliefNames</i>	<i>GetReliefResponse</i>
<i>getAllHidrogeologyNames</i>	<i>getAllHidrogeologyNames</i>	<i>GetHidrogeologyResponse</i>
<i>getAllShireNames</i>	<i>getAllShireNames</i>	<i>GetShiresResponse</i>
<i>getLessonCityNames</i>	<i>getLessonCityNames</i>	<i>GetCitiesResponse</i>
<i>getLessonReliefNames</i>	<i>getLessonReliefNames</i>	<i>GetReliefResponse</i>
<i>getLessonHidrogeologyNames</i>	<i>getLessonHidrogeologyNames</i>	<i>GetHidrogeologyResponse</i>
<i>getLessonShireNames</i>	<i>getLessonShireNames</i>	<i>GetShiresResponse</i>

A táblázatban feltüntettem a metódusok által visszaadott válaszbjektumok osztályait is. Ezen osztályok implementációja a *SOACCommonDefinition* projektben található. (Lásd Függelék 7.1 fejezet!)

Az alábbiakban ismertem az egyes funkciókat:

- ▶ **getUserName:** Ez a funkció a felhasználó nevének lekérdezéséhez használható.
- ▶ **getLearnerGroupStudents:** Ez a funkció egy adott azonosítóval rendelkező tanulócsoporthoz tartozó diákok adatainak lekérdezéséhez használható.
- ▶ **getAllCityNames:** Ez a funkció az adatbázisban tárolt települések neveinek lekérdezéséhez használható.
- ▶ **getAllReliefNames:** Ez a funkció az adatbázisban tárolt domborzati elemek neveinek a lekérdezéséhez használható.
- ▶ **getAllHidrogeologyNames:** Ez a funkció az adatbázisban tárolt vízrajzi elemek neveinek a lekérdezéséhez használható.
- ▶ **getAllShireNames:** Ez a funkció az adatbázisban tárolt megyék neveinek lekérdezéséhez használható.
- ▶ **getLessonCityNames:** Ez a funkció egy adott azonosítóval rendelkező leckében számonkért települések neveinek a lekérdezéséhez használható.
- ▶ **getLessonReliefNames:** Ez a funkció egy adott azonosítóval rendelkező leckében számonkért domborzati elemek neveinek lekérdezéséhez használható.

- ▶ **getLessonCityNames:** Ez a funkció egy adott azonosítóval rendelkező leckében számonkért vízrajzi elemek neveinek lekérdezéséhez használható.
- ▶ **getLessonCityNames:** Ez a funkció egy adott azonosítóval rendelkező leckében számonkért megyék neveinek a lekérdezéséhez használható.

4.6.8. Az AdminFunctions osztály (■)

Ez az osztály jelenleg egyetlen metódust (funkciót) sem tartalmaz. Itt lehet implementálni olyan funkciókat, amelyeket az alkalmazás adminisztrátor jogosultsági csoportjába tartozó felhasználók hívhatnak meg.

4.6.9. A TeacherFunctions osztály (■)

Ebben az osztályban vannak implementálva azok a funkciók, amelyeket az alkalmazás tanár jogosultsági csoportjába tartozó felhasználók hívhatnak meg. A kérésazonosítók és a funkciókat implementáló metódusok neveinek megfeleltetését a következő táblázat tartalmazza:

Kérésazonosító	Metódus neve	A válaszobjektum osztálya
<i>createLesson</i>	<i>createLesson</i>	<i>CreateLessonReponse</i>
<i>getTeacherLessons</i>	<i>getLessonsInfo</i>	<i>GetLessonsInfoResponse</i>
<i>getTeacherLearnerGroups</i>	<i>getLearnerGroupsInfo</i>	<i>GetLearnerGroupsInfoResponse</i>
<i>createTask</i>	<i>createTask</i>	<i>CreateTaskResponse</i>
<i>getTeacherTasks</i>	<i>getTasksInfo</i>	<i>GetTasksInfoResponse</i>

Az alábbiakban ismertem az egyes funkciókat:

- ▶ **createLesson:** Ez a funkció egy lecke létrehozásához használható.
- ▶ **getTeacherLessons:** Ez a funkció egy adott felhasználói azonosítóval rendelkező tanár által létrehozott leckék lekérdezéséhez használható.
- ▶ **getTeacherLearnerGroups:** Ez a funkció egy adott felhasználói azonosítóval rendelkező tanár tanulócsoportjainak a lekérdezéséhez használható.
- ▶ **createTask:** Ez a funkció egy gyakorlat létrehozásához használható.
- ▶ **getTeacherTasks:** Ez a funkció egy adott felhasználói azonosítóval rendelkező tanár által létrehozott gyakorlatok lekérdezéséhez használható.

4.6.10. A StudentFunctions osztály (■)

Ebben az osztályban vannak implementálva azok a funkciók, amelyeket az alkalmazás tanuló jogosultsági csoportjába tartozó felhasználók hívhatnak meg. A kérésazonosítók és a funkciókat implementáló metódusok neveinek megfeleltetését a következő táblázat tartalmazza:

Kérésazonosító	Metódus neve	A válaszbjektum osztálya
<i>getStudentLearnerGroups</i>	<i>getLearnerGroupsInfo</i>	<i>GetLearnerGroupsInfoResponse</i>
<i>getStudentTasks</i>	<i>getTasksInfo</i>	<i>GetTasksInfoResponse</i>

Az alábbiakban ismertem az egyes funkciókat:

- ▶ **getStudentLearnerGroups:** Ez a funkció azoknak a tanulócsoportoknak a lekérdezésére használható, amelyeknek az adott felhasználói azonosítóval rendelkező tanuló tagja.
- ▶ **getStudentTasks:** Ez a funkció egy adott felhasználói azonosítóval rendelkező tanuló részére létrehozott gyakorlatok lekérdezéséhez használható.

4.6.11. A SOACCommonFacadeBean osztály (■)

Az alkalmazás működése során az adatbázisban tárolt adatok lekérdezését és feldolgozását (objektumokba való becsomagolását) egy külön alrendszer végzi. A *SOACCommonFacadeBean* osztály ehhez az alrendszerhez biztosít hozzáférést a *Facade tervezési minta* alapján. Az alrendszerhez a kliensek (a *CommonFunctions*, *AdminFunctions*, *TeacherFunctions*, *StudentFunctions* osztályok és a *GatewayServlet* osztály objektumai) a Facade-n (a *SOACCommonFacadeBean* objektumon) keresztül férnek hozzá, így a Facade egy magasabb szintű interfészt szolgáltat a kliensek részére. A klienseknek nem kell ismerniük az alrendszer minden egyes osztályának a belső szerkezetét, felépítését, elég, ha a Facade által nyújtott szolgáltatásokat (metódusokat) ismerik. A Facade tervezési minta segítségével tulajdonképpen „egy egész alrendszert egyetlen objektumként ábrázolunk”. A Facade tervezési mintáról [3]-ban olvashat bővebben. A *SOACCommonFacadeBean* osztály objektumai állapotmentes session EJB objektumok, amelyek RMI/IIOP protokollon keresztül távolról is meghívhatóak.

4.6.12. A CommonServices osztály ()

Ebben az osztályban vannak implementálva a *CommonFunctions* osztályban implementált funkciókat kiszolgáló szolgáltatások (metódusok), illetve azok a szolgáltatások, amelyeket több különböző osztályba tartozó funkció is használ. A következő táblázat a *CommonServices* osztályban található szolgáltatásokat foglalja össze:

Szolgáltatás neve a Facade-ben	A bemenő objektum osztálya	A válaszbjektum osztálya
<i>getUsersInfo</i>	<i>GetUsersInfoRequest</i>	<i>GetUsersInfoRequest</i>
<i>getCities</i>	<i>GetCitiesRequest</i>	<i>GetCitiesResponse</i>
<i>getRelief</i>	<i>GetReliefRequest</i>	<i>GetReliefResponse</i>
<i>getHidrogeology</i>	<i>GetHidrogeologyRequest</i>	<i>HidrogeologyResonse</i>
<i>getShires</i>	<i>GetShiresRequest</i>	<i>GetShiresResponse</i>
<i>getLearnerGroupsInfo</i>	<i>GetLearnerGroupsInfoRequest</i>	<i>GetLearnerGropusInfoResponse</i>
<i>getTaskInfo</i>	<i>GetTasksInfoRequest</i>	<i>GetTasksInfoResponse</i>

Az alábbiakban ismertem az egyes szolgáltatásokat:

- ▶ **getUsersInfo:** Ez a szolgáltatás egy vagy több felhasználó adatainak lekérdezéséhez használható. A szolgáltatást hívó funkciók: *getUserName (CommonFunctions)*, *getLearnerGroupStudents (CommonFunctions)*
- ▶ **getCities:** Ez a szolgáltatás egy vagy több település adatainak lekérdezéséhez használható. A *getCities* szolgáltatást használó funkciók: *getAllCityNames (CommonFunctions)*, *getLessonCityNames (CommonFunctions)*
- ▶ **getRelief:** Ez a szolgáltatás egy vagy több domborzati elem adatainak lekérdezéséhez használható. A szolgáltatást hívó funkciók: *getAllReliefNames (CommonFunctions)*, *getLessonReliefNames (CommonFunctions)*
- ▶ **getHidrogeology:** Ez a szolgáltatás egy vagy több vízrajzi elem adatainak lekérdezéséhez használható. Használó funkciók: *getAllHidrogeologyNames (CommonFunctions)*, *getLessonHidrogeologyNames (CommonFunctions)*
- ▶ **getShires:** Ez a szolgáltatás egy vagy több megye adatainak lekérdezéséhez használható. A *getShires* szolgáltatást használó funkciók: *getAllShireNames (CommonFunctions)*, *getLessonShireNames (CommonFunctions)*
- ▶ **getLearnerGroupsInfo:** Ez a szolgáltatás tanulócsoportok adatainak lekérdezéséhez használható. A *getLearnerGroupsInfo* szolgáltatást használó funkciók: *getLearnerGroupsInfo (TeacherFunctions)*, *getLearnerGroupsInfo (StudentFunctions)*
- ▶ **getTasksInfo:** Ez a szolgáltatás gyakorlatok adatainak lekérdezéséhez használható. A szolgáltatást használó funkciók: *getTasksInfo (TeacherFunctions)*, *getTasksInfo (StudentFunctions)*

4.6.13. Az AdminServices osztály ()

Ez az osztály jelenleg egyetlen metódust sem tartalmaz. Itt lehet implementálni olyan szolgáltatásokat, amelyek kizárólag az *AdminFunctions* osztályban található funkciókat szolgálnak ki.

4.6.14. A TeacherServices osztály ()

Ebben az osztályban vannak implementálva azok a szolgáltatások, amelyek kizárólag a *TeacherFunctions* osztályban implementált funkciókat szolgálnak ki. A *TeacherServices* osztályban található szolgáltatásokat a következő táblázat foglalja össze:

Szolgáltatás neve a Facade-ben	A bemenő objektum osztálya	A válaszbjektum osztálya
<i>createLesson</i>	<i>CreateLessonRequest</i>	–
<i>getLessonsInfo</i>	<i>GetLessonsInfoRequest</i>	<i>GetLessonsInfoResponse</i>
<i>createLearnerGroup</i>	<i>CreateLearnerGroupRequest</i>	–
<i>createTask</i>	<i>CreateTaskRequest</i>	–

Az alábbiakban ismertem az egyes szolgáltatásokat:

- ▶ **createLesson:** Ez a szolgáltatás egy lecke létrehozásához használható. A szolgáltatást hívó funkció: *createLesson*
- ▶ **getLessonsInfo:** Ez a szolgáltatás lecek adatainak a lekérdezéséhez használható. A szolgáltatást használó funkció: *getLessonsInfo*
- ▶ **createLearnerGroup:** Ez a szolgáltatás egy tanulócsoport létrehozásához használható. A szolgáltatást jelenleg nem használja egyetlen funkció sem.
- ▶ **createTask:** Ez a szolgáltatás egy gyakorlat létrehozásához használható. A szolgáltatást hívó funkció: *createTask*

4.6.15. A StudentServices osztály ()

Jelenleg egyetlen metódust sem tartalmaz. Ebben az osztályban lehet implementálni olyan szolgáltatásokat, amelyek kizárólag a *StudentFunctions* osztályban található funkciókat szolgálnak ki.

4.6.16. A *GatewayServlet* osztály (■)

A topográfia alkalmazás felől érkező HTTP kéréseket egy külön osztály, a *GatewayServlet* osztály objektumai dolgozzák fel. Egy *GatewayServlet* objektum a topográfia alkalmazás által küldött HTTP kérés megválaszolásához szükséges adatbázisbeli adatokat szintén egy *SOACommonFacadeBean* példányon keresztül éri el.

4.7. A topográfia alkalmazás felépítése, működése

A topográfiai alkalmazás egy Java kisalkalmazás, úgynevezett *applet*, amely a felhasználó számítógépén, a böngésző *ClassLoader*-e alatt kerül betöltésre és futtatásra. Az alkalmazás a webes felületen keresztül a keretalkalmazásból indítható. Az *applet* (a *com.dns.client.Application* osztály egy példánya) elindítását követően annak *init* metódusa kapja meg a vezérlést*. Mivel a felhasználók bejelentkezése és a gyakorlatok elindítása a webes felületen keresztül történik, így a munkamenet azonosítót és a kiválasztott gyakorlat azonosítóját az *applet* paraméterként kapja meg. Az *init* metódus elvégzi a program paramétereinek a feldolgozását és a felhasználói felület összeállítását, majd az *applet* *paint* metódusa megrajzolja a felhasználói felületet. Az alkalmazás grafikus felhasználói felületét saját komponensek építik fel. (Lásd 4.7.1 fejezet!) Minden komponens az általa elfoglalt képernyőterület megrajzolását végzi. A saját komponensek osztálydiagramját a 5. ábra szemlélteti.

Az alkalmazás működése eseményvezérelt, azaz a program futásának menete nem rögzített, annak alakulását a felhasználók által kiváltott külső események határozzák meg. Ilyen esemény lehet például egy gombnak a lenyomása vagy egy menüpontnak a kiválasztása, stb. Az *applet* elindítását követően az alkalmazás komponensei készen állnak a felhasználói események fogadására és feldolgozására.

* Amikor egy böngészőprogram futtat egy Java *applet*-et, annak végrehajtása az *applet* *public void init();* metódusával kezdődik. Az *init* metódusnak hasonló a rendeltetése, mint a C/C++ nyelvek *main* függvényének, a *main* függvénytől eltérően azonban az *applet* futása nem fejeződik be akkor, amikor az *init* függvény futása befejeződik.

Ha a felhasználó a paraméterként megkapott gyakorlat elindításához az alkalmazás *Indítás* gombjára kattint, akkor az applet az elindított gyakorlat adatainak megszerzése érdekében kéréseket küld az alkalmazáservernek. Egy kérés elküldéséhez az applet a *com.dns.client.communication.AppContext* osztály egy objektumának

```
public synchronized void doRequest(String requestId);
```

metódusát használja. Egy adott kérés alkalmával átküldött adatok és azonosító információk (pl. IP cím, munkamenet azonosító, stb.) a kérés elküldése előtt, a kliens oldalon egy speciális objektumba, a *com.dns.common.communication.SimpleData* osztály egy példányába kerülnek. Egy *SimpleData* objektum tulajdonképpen egy általános kérés objektum, amelybe az applet által küldhető bármely adat becsomagolható, és amely minden szükséges információt tartalmaz az adott kérésre, illetve kommunikációra vonatkozóan. Amikor az applet kéréseket intéz a kiszolgálóhoz, akkor tulajdonképpen ilyen kérés objektumok kerül átvitelre. A server és az alkalmazás között minden *SimpleData* objektum egy-egy HTTP üzenet törzsében utazik. Az applet az objektumokat feltölti adatokkal, majd HTTP kérések törzsébe csomagolva átküldi azokat a kiszolgálónak. A kérések elküldése és kiszolgálása sorosan történik. Amennyiben egy kérés kiszolgálása közben újabb igény érkezik a kommunikációt vezérlő objektumhoz (*AppContext*), akkor az új kérés egy várakozási sorba kerül.

Az alkalmazás által elküldött HTTP kéréseket a *GatewayServlet* osztály objektumai dolgozzák fel. A feldolgozó *GatewayServlet* példány a *SimpleData* objektumban érkezett adatokat átcsomagolja a kérésazonosítónak megfelelő szolgáltatás bemenő objektumába, majd azzal meghívja a Facade, azaz egy *SOACommonFacadeBean* objektum megfelelő metódusát. A kérésazonosítót a szervlet szintén a *SimpleData* objektumba csomagolva kapja meg. Az applet által küldhető kérések azonosítóit és az általuk elérhető szolgáltatások listáját az alábbi táblázat foglalja össze:

Kérésazonosító	Szolgáltatás neve a Facade-ben
<i>getTaskInfo</i>	<i>getTasksInfo</i>
<i>getCities</i>	<i>getCities</i>
<i>getRelief</i>	<i>getRelief</i>
<i>getHidrogeology</i>	<i>getHidrogeolody</i>
<i>getShires</i>	<i>getShires</i>
<i>saveResult</i>	<i>saveResult</i>

A következőkben ismertem az egyes kéréseket:

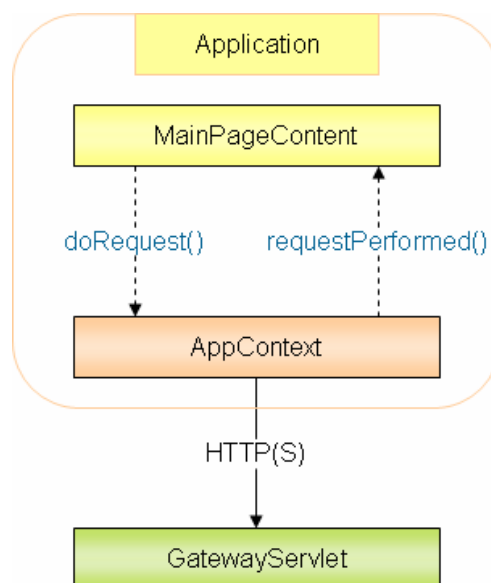
- ▶ **getTaskInfo:** Ez a kérés egy adott azonosítóval rendelkező gyakorlat adatainak a lekérdezéséhez használható.
- ▶ **getCities:** Ez a kérés egy adott azonosítóval rendelkező gyakorlatban számonkért települések adatainak a lekérdezéséhez használható.
- ▶ **getRelief:** Ez a kérés egy adott azonosítóval rendelkező gyakorlatban számonkért domborzati elemek információinak a lekérdezéséhez használható.
- ▶ **getHidrgelology:** Ez a kérés egy adott azonosítóval rendelkező gyakorlatban számonkért vízrajzi elemek adatainak a lekérdezéséhez használható.
- ▶ **getShires:** Ez a kérés egy adott azonosítóval rendelkező gyakorlatban számonkért megyék információinak a lekérdezéséhez használható.
- ▶ **saveResult:** Ez a kérés egy adott azonosítóval rendelkező gyakorlat megoldása során elért eredmény elmentéséhez használható.

A *GatewayServlet* szervletpéldány által visszaküldött válasz szintén egy *SimpleData* osztályú objektum, egy HTTP válaszüzenet törzsébe csomagolva. A válasz megérkezését követően az applet *com.dns.client.content.page.MainPageContent* komponensének

```
public void requestPerformed(String requestId);
```

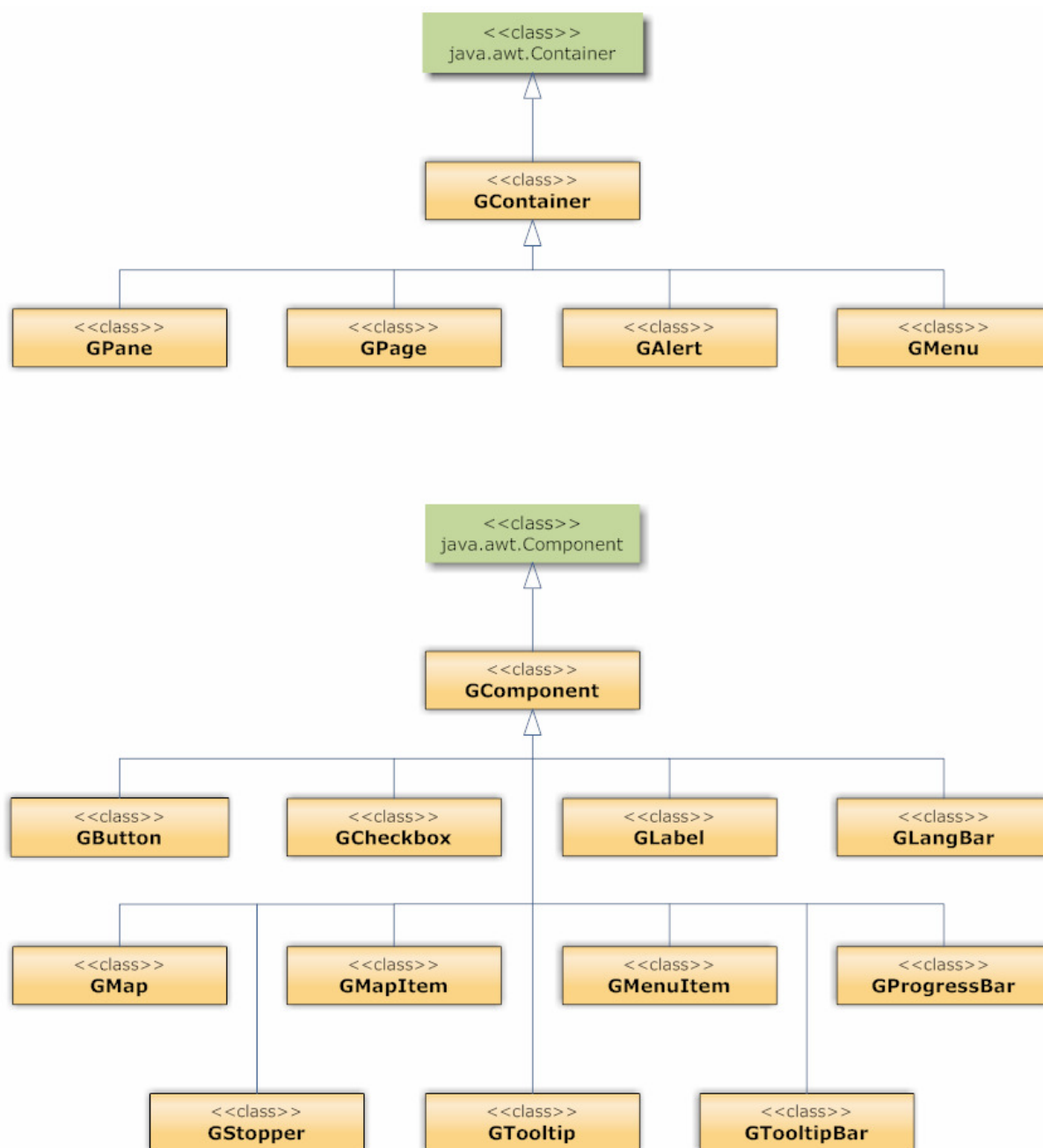
metódusa kapja meg a vezérlést, amely elvégzi az adatok feldolgozását és a komponensek frissítését, vagyis a megjelenítést.

Az alkalmazás és az alkalmazásszerveren futó *GatewayServlet* példány kommunikációját a következő ábra szemlélteti:



4.7.1. Saját komponensek

Az alkalmazás grafikus felhasználói felülete saját komponensekből épül fel. A saját komponensek osztálydiagramját a következő ábra szemlélteti:



5. ábra Komponensek osztálydiagramja

Az 5. ábrán látható saját komponensek a *com.dns.client.glt.com.GComponent*, illetve a *com.dns.client.glt.com.GContainer* osztályok leszármazottai. A *GComponent* osztály a *java.awt.Component*, a *GContainer* osztály pedig a *java.awt.Container* osztály közvetlen leszármazottja. A továbbiakban a *GContainer* osztály leszármazottait, megkülönböztetésül a *GComponent* osztályból származtatott komponensektől, konténereknek hívom. A konténerek olyan komponensek, amelyek több komponenset fognak össze egyetlen komponenssé. Ily módon felépíthető egy komponensfa, amelynek levelei komponensek, levéltől különböző csomópontjai pedig konténer objektumok.

A felhasználóval folytatott kommunikáció szempontjából minden komponens kétféle állapotban lehet, vagy engedélyezett, azaz részt vehet a felhasználóval folytatott interakcióban, vagy letiltott, azaz nem válaszolhat a bekövetkező külső eseményekre. A konténer és komponens objektumok állapotának beállításához az adott objektum

```
public void setEnabled(boolean enabled);
```

metódusát használhatjuk, amely a *GComponent*, illetve *GContainer* osztályokban van implementálva.

Ha egy konténer számára tiltott a felhasználóval folytatott kommunikáció, akkor az minden olyan komponens és konténer számára is tiltott, amelyeknek ő a komponensfában szülő konténer. Az állítás megfordítása nem szükségszerűen igaz, vagyis egy engedélyezett állapotú konténernek lehetnek letiltott gyermekei. Ha azonban egy komponens engedélyezett, akkor elmondható, hogy egyetlen szülő konténer sem lehet letiltott. Ezen viselkedésminta teljesülését az őszosztályokban (*GContainer*, *GComponent*) implementált *setEnabled* metódusok biztosítják.

Egy szülő konténer ismeri a gyermekeinek az állapotát. Amikor egy adott konténer letiltásra kerül, akkor elkészít és letárol egy állapotterképet mindazon objektumok állapotáról, amelyeknek ő szülő konténer a komponenshierarchiában, majd ezt követően minden gyermekobjektum állapotát letiltott állapotra állítja. Ha az adott konténer alkonténereket is tartalmaz, akkor az állapotváltás algoritmusuk rekurzívan végighalad az adott alkonténer által kijelölt részén. Az állapotterkép tulajdonképpen egy „pillanatkép”, amely a konténer letiltásának pillanatában eltárolja annak „belső állapotát”, vagyis az általa tartalmazott

gyermekobjektumok állapotát. Erre az eredeti, letiltás előtti állapotjellemezők visszaállítása miatt van szükség, amikor ugyanis az adott konténer állapota a program futása során ismét engedélyezetté válik, akkor a konténer az állapottérkép segítségével vissza tudja állítani a gyermekobjektumok eredeti állapotát. Egy tipikus gyakorlati példája, felhasználási területe a mechanizmus működésének a következő: egy HTTP kérés indítása után, a válasz megérkezéséig az applet minden konténerre és komponensre letiltásra kerül, a válasz megérkezését és feldolgozását követően, azonban vissza kell állítani a komponensek eredeti állapotát. Ebben az esetben a letiltást és ismételt engedélyezést szükségtelen egy ciklusban az applet minden komponensére elvégezni, elegendő a komponensfa gyökérelemének állapotát megváltoztatni.

Néha azonban az is szükségessé válhat, hogy egy komponens állapota az őt tartalmazó konténer állapotának engedélyezését követően a régi, letiltás előtti állapottól eltérő állapotot tükrözzön, azaz szeretnénk, hogy egy adott komponens állapota megváltozhasson az őt tartalmazó konténer letiltott állapota esetén is. Ez az állapotváltozás azonban a fentebb leírt alapértelmezett viselkedésmód sértetlenségének fenntartása mellett nyilvánvalóan csak a konténer engedélyezését követően léphet életbe. Az említett két ősztyály *setEnabled* metódusai ezeket a kívánalmakat is kielégítik.

A következőkben megadom az ősztyályokban implementált *setEnabled* metódusok forráskódját.

A *GComponent* ősztyály *setEnabled* metódusának implementációja:

```
1. public void setEnabled(boolean enabled) {
2.     Container parent = getParent();
3.     if (parent != null && !parent.isEnabled()) {
4.         if (parent instanceof GContainer) {
5.             Vector<Component> vComponents =
6.                 ((GContainer)parent).vDisabledComponents;
7.             if (enabled) {
8.                 if (!vComponents.contains(this))
9.                     vComponents.addElement(this);
10.            }
11.            else if (vComponents.contains(this))
12.                vComponents.removeElement(this);
13.        }
14.    }
15.    else
16.        this.enabled = enabled;
17. }
```

A *GContainer* osztály *setEnabled* metódusának forráskódja:

```
1. public void setEnabled(boolean enabled) {
2.     // az esetleges tartalmazó kontener lekerdezesere es tarolasa
3.     Container parent = getParent();
4.
5.     // ha a kontenernek van tartalmazó kontenere, es annak
6.     // allapota letiltott
7.     if (parent != null && !parent.isEnabled()) {
8.         // ha a tartalmazó kontener a GContainer osztaly
9.         // leszarmazottja, akkor az állapotvaltas tarolásra
10.        // kerül, es a tartalmazó kontener engedelyezesekor
11.        // aktualizalodik ezen kontener allapota is
12.        if (parent instanceof GContainer) {
13.            Vector<Component> vComponents =
14.                ((GContainer)parent).vDisabledComponents;
15.            if (enabled) {
16.                if (!vComponents.contains(this))
17.                    vComponents.addElement(this);
18.            }
19.            else if (vComponents.contains(this))
20.                vComponents.removeElement(this);
21.        }
22.    }
23.    // ha a kontenernek nincs tartalmazó kontenere, vagy van,
24.    // de annak allapota engedelyezett
25.    else {
26.        // ha nem tortenik tenyleges állapotvaltas
27.        if (this.enabled == enabled)
28.            return;
29.
30.        if (enabled) { // ha a kontenert engedelyezzuk
31.            this.enabled = enabled;
32.            for(int i = 0; i < vDisabledComponents.size(); i++)
33.                vDisabledComponents.elementAt(i).setEnabled(true);
34.            vDisabledComponents = new Vector<Component>();
35.        }
36.        else { // ha a kontenert letiltjuk
37.            Component [] components = getComponents();
38.            for (int i = 0; i < components.length; i++)
39.                if (components[i].isEnabled()) {
40.                    if (components[i] instanceof GTooltip)
41.                        ((GTooltip)components[i]).hide();
42.                    components[i].setEnabled(false);
43.                    vDisabledComponents.addElement(components[i]);
44.                }
45.            this.enabled = enabled;
46.        }
47.    }
48. }
```

A *com.dns.client.glt.com.event* csomag saját eseménykezelőket és eseményosztályokat tartalmaz. Egy saját komponenshez egyidejűleg több azonos típusú eseményfigyelő (listener) objektum is hozzárendelhető.

4.7.2. Hibakeresés

Az applet a fejlesztési idejű hibakeresést és tájékoztatást segítő különböző típusú információkat jeleníthet meg a Java konzolon. A konzolon megjeleníthető információk két nagy csoportba sorolhatóak, a tájékoztató jellegű bejegyzések és a hibák csoportjába. A megjelenített naplóbejegyzések típusát, azaz a naplózás szintjét az applet paraméterként kapja meg. Minden bejegyzés egy időbélyegzővel kezdődik, amelyet a típusra vonatkozó rövid információ (INFO, ERROR), és végül a bejegyzés szövege követ. Ha a naplózás engedélyezett, akkor a konzolon kaphatunk információt többek között például az elküldött kérés és a szervertől visszakapott válasz objektumokban tárolt adatok számáról, értékéről, stb. Ezen információk kinyeréséhez a *GENERAL REQUEST* és a *GENERAL RESPONSE* naplóbejegyzések értékeit kell megvizsgáljunk.

```
Java Console
Java Plug-in 1.5.0_05
Using JRE version 1.5.0_05 Java HotSpot(TM) Client VM
User home directory = C:\Documents and Settings\admin_

-----
c: clear console window
f: finalize objects on finalization queue
g: garbage collect
h: display this help message
l: dump classloader list
m: print memory usage
o: trigger logging
p: reload proxy configuration
q: hide console
r: reload policy configuration
s: dump system and deployment properties
t: dump thread list
v: dump thread stack
x: clear classloader cache
0-5: set trace level to <n>
-----

[Sat Jan 10 03:01:19 CET 2009] INFO: GToolTip[txt:mainpagecontent.tooltip.tip_01|enabled:true] show
[Sat Jan 10 03:03:30 CET 2009] INFO: GLangBar[lang_code:EN|enabled:true] state changed
[Sat Jan 10 03:03:32 CET 2009] INFO: GLangBar[lang_code:HU|enabled:true] state changed
[Sat Jan 10 03:03:35 CET 2009] INFO: GCheckbox[label:mainpagecontent.checkbox.shires|enabled:true|checked:false] state changed
[Sat Jan 10 03:03:36 CET 2009] INFO: GCheckbox[label:mainpagecontent.checkbox.shires|enabled:true|checked:true] state changed
[Sat Jan 10 03:03:38 CET 2009] INFO: GCheckbox[label:mainpagecontent.checkbox.relief|enabled:true|checked:false] state changed
[Sat Jan 10 03:03:44 CET 2009] INFO: GLangBar[lang_code:EN|enabled:true] state changed
[Sat Jan 10 03:03:59 CET 2009] INFO: GCheckbox[label:mainpagecontent.checkbox.relief|enabled:true|checked:true] state changed
[Sat Jan 10 03:04:09 CET 2009] INFO: GLangBar[lang_code:HU|enabled:true] state changed
[Sat Jan 10 03:04:15 CET 2009] INFO: GLangBar[lang_code:EN|enabled:true] state changed
[Sat Jan 10 03:05:42 CET 2009] INFO: GMenuItem[selected item:GMenuItem[label:mainpagecontent.menuitem.start|enabled:true|checked:false]] state changed
[Sat Jan 10 03:05:43 CET 2009] INFO: GENERAL REQUEST (getCities) := [{examId=1}]
[Sat Jan 10 03:05:44 CET 2009] INFO: GENERAL RESPONSE := [{GetCitiesResponseItem={northernLength=19:03:56, name=Budapest, easternWidth=47:29:95}, {northernLength=21:38:23, name=Debrecen}}]
[Sat Jan 10 03:05:44 CET 2009] INFO: GENERAL REQUEST (getRelief) := [{examId=1}]
[Sat Jan 10 03:05:45 CET 2009] INFO: GENERAL RESPONSE := [{GetReliefResponseItem={regionCode=#fffff, name=Bukk}, {regionCode=#fffff, name=Mecsek}}]
[Sat Jan 10 03:05:46 CET 2009] INFO: GENERAL REQUEST (getHydrogeology) := [{examId=1}]
[Sat Jan 10 03:05:47 CET 2009] INFO: GENERAL RESPONSE := [{GetHydrogeologyResponseItem={regionCode=#ff8000, name=Szamos}, {regionCode=#007cc3, name=Tisza}, {regionCode=#ff0000, name=Zemplén}}]
[Sat Jan 10 03:05:47 CET 2009] INFO: GENERAL REQUEST (getShires) := [{examId=1}]
[Sat Jan 10 03:05:48 CET 2009] INFO: GENERAL RESPONSE := [{GetShiresResponseItem={regionCode=#fefefe, name=Szabolcs-Szatmár-Bereg megye}, {regionCode=#fdfdfd, name=Borsod-Abaúj-Zemplén}}]
[Sat Jan 10 03:05:49 CET 2009] INFO: GStopper[time:0] start
[Sat Jan 10 03:05:49 CET 2009] INFO: GToolTip[txt:mainpagecontent.tooltip.tip_02|enabled:false] show
[Sat Jan 10 03:06:58 CET 2009] INFO: GToolTipBar[checked:true] state changed
[Sat Jan 10 03:06:58 CET 2009] INFO: GToolTip[txt:mainpagecontent.tooltip.tip_02|enabled:true] show
[Sat Jan 10 03:06:58 CET 2009] INFO: GToolTipBar[checked:false] state changed
[Sat Jan 10 03:07:02 CET 2009] INFO: GLangBar[lang_code:HU|enabled:true] state changed
```

6. ábra

Részlet a Java konzolon megjelenő naplóból.

5. ÖSSZEFOGLALÁS

Az elkészített és a dolgozatban bemutatott alkalmazás egy olyan oktató- és készségfejlesztő program, amelyet Magyarország természetföldrajzi és településföldrajzi topográfiájának oktatásához, a tanórai oktatás kiegészítéseként, a tanítási-tanulási folyamat támogatásához készítettem. Bár az alkalmazás a bevezetőben megfogalmazott célokat megvalósítja, számos továbbfejlesztési igény támasztható vele szemben. A program jelen verziója például kevés funkcióval rendelkezik. Nagy hiányossága, hogy az adminisztrátori funkciók egyáltalán nem lettek implementálva. Ezen funkciók kifejlesztése, illetve a funkciók számának bővítése képezheti a továbbfejlesztés fő irányát.

A program ugyanakkor Magyarország természetföldrajzának és településföldrajzának oktatása során jól használható tájékozási feladatok összeállítására és tanítására, de számos egyéb földrajzi, sőt topográfiai feladat megoldására nem képes. Ez, illetve több interaktív vaktérkép elkészítése ugyancsak képezhetné a továbbfejlesztés egy-egy lehetséges irányát.

IRODALOMJEGYZÉK

- [1] Nyékyné Gaizler Judit: *JAVA 2 Útikalauz programozóknak 1.3 I-II.*, ELTE TTK Hallgatói Alapítvány, Budapest, 2001, ISBN 963 463 486 9
- [2] Nyékiné Gaizler Judit: *J2EE Útikalauz Java programozóknak*, ELTE TTK Hallgatói Alapítvány, Budapest, 2002, ISBN 9789634635784
- [3] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Programtervezési minták*, Kiskapu Kft., Budapest, 2004, ISBN 963 9301 77 9
- [4] Michael Morrison: *Tanuljuk meg az XML használatát 24 óra alatt*, Kiskapu Kft., Budapest, 2006, ISBN 9639637092
- [5] Tim Lindholm, Frank Yellin: *The Java Virtual Machine Specification*, Second Edition, 1997
- [6] Markus Dahm: *Byte Code Engineering*, Institut für Informatik, Freie Universität Berlin
- [7] Markus Dahm: *Byte Code Engineering with the BCEL API*, Technical Report B-17-98, Institut für Informatik, Freie Universität Berlin, April 3, 2001
- [8] Christian Collberg, Clark Thomborson, Douglas Low: *A Taxonomy of Obfuscating Transformations*, Technical Report #148, Department of Computer Science, The University of Auckland, 1997
- [9] Christian Collberg, Clark Thomborson, Douglas Low: *Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs*, SIGPLAN-SIGACT POPL'98, ACM Press, San Diego, CA, January 1998.
- [10] Christian Collberg, Clark Thomborson: *Software Watermarking: Models and Dynamic Embeddings*, Technical Report, Department of Computer Science, The University of Auckland, 1998
- [11] Christian Collberg, Clark Thomborson: *Watermarking, Tamper-Proofing, and Obfuscation – Tools for Software Protection*, Technical Report #170, Department of Computer Science, The University of Auckland; also: Technical Report 2000-03, Department of Computer Science, University of Arizona, 2000
- [12] Gregory Wroblewski: *General Method of Program Code Obfuscation*, PhD Dissertation, Wrocław University of Technology, Institute of Engineering Cybernetics, 2002

FÜGGELÉK

7.1. Projektek szervezése az Eclipsben

Ebben a fejezetben megadom az alkalmazás fejlesztése során létrehozott Eclipse projektek listáját, és röviden ismertetem azok funkcióját, szerepét.

Az alkalmazás fejlesztése során létrehozott Eclipse projektek:

- ▶ **CommonCodeBase:** A program működése során felhasznált alapeszközök osztályait tartalmazó projekt. Ebben a projektben kaptak helyet pl. a sablonkezelő osztályok, stb.
- ▶ **CommonTagLib:** Ez a projekt tartalmazza a JSP oldalak írásához használt közös JSP akcióelemek elemkezelő Java osztályait, mint pl. a JSP oldalak által generált XHTML kód „homályosítását” végző `<obfuscate></obfuscate>` tag `com.dns.common.jsp.handler.ObfuscateTag` elemkezelő osztálya, stb.
- ▶ **SOACommonDefinition:** A szolgáltatás alrendszer szolgáltatásainak ki-és bemenő típusait tartalmazó projekt.
- ▶ **SOAPrimitiveBase:** Itt található az adatbázis kommunikációért felelős csatoló (*connector*) objektumok osztálya.
- ▶ **SOACommonBusiness:** A szolgáltatás alrendszer osztályainak projektje.
- ▶ **SOAFacadeBase:** A Facade implementálásához felhasznált alaposztályok kaptak helyet ebben a projektben.
- ▶ **SOACommonFacadeEJB:** A Facade implementációját tartalmazó projekt.
- ▶ **SOACommonFacadeEJBClient:** A Facade `SOACommonFacadeBean` EJB objektumának távolról történő hívásához szükséges kliensoldali kódot tartalmazó projekt.
- ▶ **SOACommonEAR:** Az alkalmazás összeállításakor a Facade implementációját tartalmazó `SOACommonFacadeEJB` projekt, a Facade-n keresztül hívható szolgáltatások ki- és bemenő típusait tartalmazó `SOACommonDefinition` projekttel, és az alrendszer implementációját tartalmazó `SOACommonBusiness` projekttel együtt egyetlen telepíthető (deploy) EAR csomagba kerül. Ezen telepíthető archívum összeállítását (build) végző ANT szkript kapott helyet a `SOACommonEAR` projektben.

- ▶ **GatewayWAR:** A *GatewayServlet* osztályt és annak működéséhez szorosan kapcsolódó osztályokat tartalmazó projekt.
- ▶ **ClientApplet:** A topográfia alkalmazás osztályait tartalmazó projektek.
- ▶ **ClientAdminTagLib:** Ebben a projektben találhatóak a JSP oldalak írásához használt adminisztrációs feladatokat végrehajtó JSP akcióelemek elemkezelő osztályai.
- ▶ **ClientAdminWAR:** Ez a projekt tartalmazza az alkalmazás JSP oldalait, XSL stíluslapjait, a ki- és bejelentkezésért felelős objektumok osztályait, illetve az egyes funkciókat implementáló osztályokat is.
- ▶ **Utilities:** Ez a projekt az alkalmazáshoz fejlesztett hasznos segédprogramokat tartalmaz.