DIPLOMAMUNKA

Orvos Gergő Attila

Debrecen
2010

DEBRECENI EGYETEM

Informatikai Kar

Portálkészítés japán szótár és tananyagok számára PHP nyelven

Témavezető:
Dr. Horváth Géza
Egyetemi adjunktus

Készítette:
Orvos Gergő Attila
Programtervező Matematikus
Hegedüs Péter
Programtervező Informatikus
(BSc)

Debrecen
2010

Tartalomjegyzék

I. Bevezetés

Az internet jött, látott, és győzött. Megszületése óta folyamatosan terjed, és egyre nagyobb mértékben hálózza be a világot éppúgy, mint az életünket. A Netcraft becslései szerint a világhálón több, mint 205 millió weboldal található. A Központi Statisztikai Hivatal adatai alapján Magyarországon az internet-előfizetések száma 2010-re meghaladta a 2,5 milliót. Rengeteg felhasználó jut temérdek mennyiségű adathoz. Legyen szó információ keresésről, kapcsolattartásról, kikapcsolódásról, munkáról vagy vásárlásról a böngészők használata mindennapossá vált.

Jelentős változáson ment keresztül az internetes technológia is. Keretrendszerek, tervezési minták, integrált fejlesztőkörnyezetek segítik elő a hatékony programozást. Emellett számos lehetőség kínálkozik a külcsín elképzelésünk szerinti elkészítéséhez, így a tartalom mellett egyre inkább előtérbe került a forma, azaz a stílus kialakítása. A kezdeti években jellemző statikus, egyszerű megjelenésű webszájtokat jól tervezett portálok váltották fel.

A portálok dinamikusan változó tartalmú honlapok, melyek különféle szolgáltatást nyújtanak az internetezők számára. Léteznek portálépítő szoftverek, de akár magunk is nekikezdhetünk az egymáshoz kapcsolódó oldalak elkészítésének. Ez utóbbi esetben technológiát kell választanunk, amely lehet Java (JavaServer Pages), ASP (Active Server Pages) vagy PHP. Minden választásnak van előnye és hátránya egyaránt. A PHP alacsonyabb erőforrás igénye mellett kevésbé biztonságos, azonban könnyen gyorsan programozható. A JSP teljesen objektumorientált, több szabványt támogat, biztonságosabb, cserébe viszont több szerverkapacitást igényel. Az ASP-t Windows-os IIS szerverrel és MS-SQL-el szokás használni, amelyek nem ingyenesek, azonban ASP.NET használatával fejlettebb technológiát tudhatunk magunk mögött.

Diplomamunkám célja, egy olyan PHP alapú portál készítésének bemutatása, melyre fájlokat, dokumentumokat tölthet fel vagy törölhet az arra jogosult felhasználó, valamint magába foglal egy japán–magyar szótárt is. Azért esett a választás a PHP-ra, mert könnyen elsajátítható, elterjedt, sok webtárhely nyújt a futtatáshoz szükséges környezetet, valamint sok

ingyenes eszköz áll rendelkezésre a fejlesztés és futtatás kényelmesebbé tételére. A megoldandó feladatok között szerepel tehát a megvalósítandó funkciók implementálása, adatbázisterv és adatbázis létrehozás, valamint a kinézet megtervezése és megvalósítása. Mivel a program projekt munkában készül, így jelen írás nem tér ki a dizájnra, erről ugyanis Hegedüs Péter megegyező című szakdolgozatában olvashatunk bővebben. Írásomban a PHP-val történő weblap készítés és az adatbázis tervezésével, létrehozásával kapcsolatos témaköröket szándékozom körüljárni, valamint bemutatni ezekhez a feladatokhoz szükséges, többnyire ingyenes szoftvereket.

Terveim szerint a munka eredményeként létre fog jönni egy olyan váz, amely a későbbiekben alkalmas lesz arra, hogy az alapját képezze egy tetszőleges tartalommal bíró honlap elkészítésének. Ezáltal egy jó kiinduló alapot kapunk későbbi munkáinkhoz. Csupán a kimaradt vagy a szükséges plusz funkciókat kell majd implementálni, illetve a megjelenítésért felelős fájlokat lecserélni.

II. Felhasznált segédeszközök, környezetek

PHP

A PHP (PHP: Hypertext Preprocessor) egy nyílt forráskódú, általános célú programozási nyelv, úgynevezett szkriptnyelv, mely kiváló lehetőséget nyújt a dinamikus web-fejlesztéshez. Önállóan vagy HTML-be ágyazva is használható. A nyelv imperatív, lehetőséget adva objektumorientált programozásra, gyengén típusos, a típus-hozzárendelés és -ellenőrzés dinamikusan, futási időben történik. Használatát platformfüggetlensége könnyíti meg, fordítóprogram nem szükséges hozzá.

Megalkotása Rasmus Lerdorf nevéhez fűződik, aki 1995-ben tette közzé az első nyilvános változatot, amelyet Personal Home Page Tools néven ismert meg a nagyközönség. A továbbfejlesztésnek köszönhetően 1997-ben látott napvilágot a PHP/FI 2, mely a korábbi verziót adatbázissal történő kommunikációval és űrlapfeldolgozóval egészítette ki. 1998-ban Zeev Suraski és Andi Gutmans közreműködésével debütált a PHP 3-as változata. Két év múlva jelent meg a PHP 4, mely a webkiszolgálókkal történő kapcsolat egységesítését szolgáló alréteggel bővült, mely elősegítette az Apache-on kívül más szerver környezet használatát. Elkezdődött a modulok és az objektumok támogatása. Jelenleg a PHP 5 az elfogadott, támogatott verzió, amelyben jelentős javításon ment keresztül az objektumorientált eszközrendszer, de már folyamatban van a PHP 6 fejlesztése is.

PHP fájlt létrehozhatunk egyszerű jegyzettömbbel is, mégis érdemesebb szerkesztőt vagy integrált fejlesztőkörnyezetet (Integrated Development Environment) használni. Az editorok alacsony erőforrásigényű programok, melyek többféle fájl formátumot és karakterkódolást támogatnak. A beépített programnyelvi támogatás miatt képesek a kulcsszavak, kódblokkok és változók vizuális elkülönítésére. Ilyen eszközök például a Notepad++ vagy a PSPad. Fejlesztőkörnyezetek közül választhatjuk a NetBeans-t, a phpDesigner-t, vagy a Zend Studiot. Az IDE-k telepítéséhez több helyre van szükség a merevlemezen, futtatásukkor pedig nagyobb erőforrásigényeik vannak. Cserébe viszont a kód írása közben is képesek jelezni hibákat, vagy segítenek a kódgenerálásban, kiegészítésben, formázásban. A PHP állományok

alapértelmezett kiterjesztése a *.php*, ez azonban megváltoztatható a kiszolgáló beállításaiban. Nézzük meg, hogyan is néz ki, a klasszikus „Hello World!” program, ha HTML kódba szúrjuk be a PHP-t:

```
<html>
   <head>
        <title>
             Hello World in PHP
        </title>
   </head>
   <body>
        <?
             print("Hello World");
        ?>
   </body>
</html>
```

A PHP beépített adatbázis kezelő függvényekkel van ellátva, amelyek közül a legfontosabbak:

- a *mysql_connect* a kiszolgálóhoz történő csatlakozásra való. Paraméterként meg kell adnunk a szerver IP címét és a port számot, az adatbázisban regisztrált felhasználó nevét és jelszavát. Siker esetén egy kapcsolatazonosítót ad vissza.
- a *mysql_close* lezárja a kapcsolatot.
- a *mysql_select_db*-vel kiválasztjuk az adatbázist, mellyel dolgozni szeretnénk. Ehhez a nevét és egy kapcsolatazonosító értéket kell a függvény paraméterlistájában megadni.
- a *mysql_errno* és a *mysql_error* a műveletvégzés során felmerülő hibák kódjához és a hibaüzenetekhez biztosítja a hozzáférést.
- a *mysql_query* az SQL parancsok futtatását végzi. Használatával adatokat szúrhatunk be vagy kérdezhetünk le tábláinkból.
- a *mysql_fetch_row* az eredménysorok kezelését végzi.

MySQL

A MySQL egy SQL (Structured Query Language – Strukturált Lekérdező Nyelv) alapú relációs adatbázis-kezelő rendszer, amely többfelhasználós és többszálú támogatást nyújt. Sok

platformon és a legtöbb programnyelvből illesztőfelületekkel könnyen elérhető. Nyílt forráskódú, így rendkívül költséghatékony megoldást jelent, webfejlesztők számára.

A svéd MySQL AB cég fejlesztette, amelyet 2008-ban a Sun Microsystems felvásárolt, mely 2010 óta az Oracle Corporation leányvállalata.

Az SQL nyelv és így a MySQL utasításai három nagy alcsoportra oszthatók, melyek a következők:

- Adatdefiníciós nyelv (Data Definition Language – DDL): az adatbázis valamint a táblák létrehozása, törlése, átnevezése és módosítása tartozik ide;
- Adatmanipulációs nyelv (Data Manipulation Language – DML): az adatok beszúrásához, törléséhez, módosításához és lekérdezéséhez szükséges utasításokat foglalja magába;
- Adatvezérlő nyelv (Data Control Language – DCL): a tranzakcióvezérlés és a felhasználói hozzáférés eszközrendszerét tartalmazza.

Az adatbázissal történő kommunikációt a PHP beépített függvényeivel biztosítjuk.

Apache

Az Apache HTTP Server egy nyílt forráskódú webkiszolgáló. Kompatibilis többek között a Linux, Windows, Unix, Mac OS X és Solaris platformokkal. Előnye a gyorsaság mellett, a biztonság, a fejleszthetőség, kiváló naplózási technikája, valamint a modulok formájában megvalósított tulajdonságok és a támogatott programnyelvek sokszínűsége. A PHP mellett együttműködik a Perl, Tcl és a Python nyelvekkel is. Alkalmas üzleti és magán felhasználásra egyaránt. A Netcraft adatai szerint piaci részesedése több, mint ötven százalékos. Bár a Port80 elemzése alapján kitűnik, hogy nagyvállalati környezetben inkább a Microsoft Internet Information Services programját használják, ahol az Apache részesedése körülbelül tizenöt százalékos, szemben az IIS ötvenhárom százalékával.

Az Apache fejlesztése az NCSA (National Center for Supercomputing Applications)

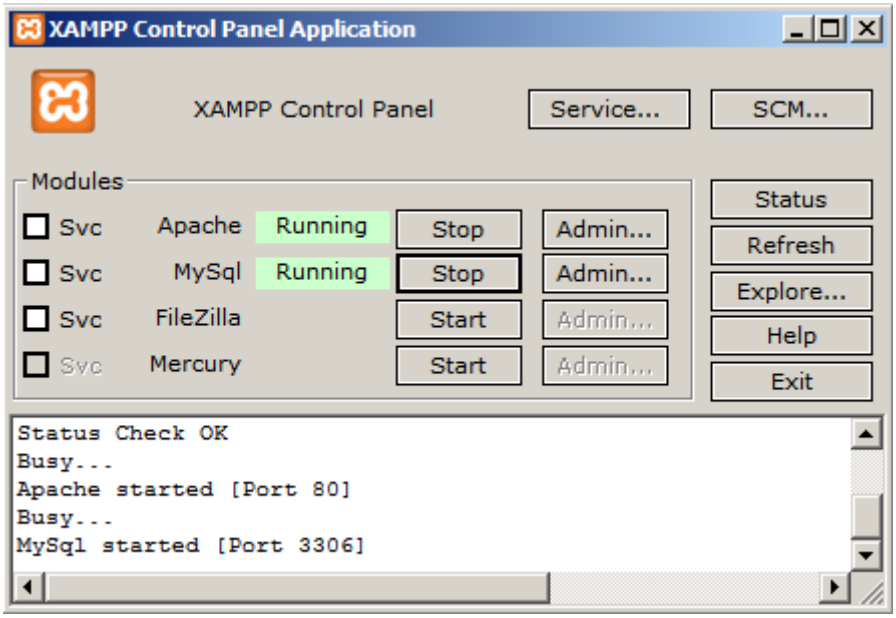
webszerver folytatásából indult 1994-ben, amely Rob McCool nevéhez fűződik. A rendszergazdák kezdetben levelezéssel juttatták el egymáshoz az általuk fejlesztett javításokat. A második kiadásban bevezetésre került a moduláris szemlélet, továbbá újraírták az első verzió nagy részét. A legfrissebb elérhető stabil változat a 2.2.15-ös számot viseli.

Az Apache működés közben tulajdonképpen a dokumentumkönyvtárában elhelyezett fájlokat adja át a csatlakozó klienseknek HTTP protokollon keresztül. A *httpd.conf* állomány szerkesztésével bármely könyvtár beállítható az alapértelmezett munkakönyvtár helyett. Működése a PHP szempontjából igen egyszerű. Ha a beállított dokumentumkönyvtár tartalmaz *index.php* állományt, akkor megnyitja azt és a leprogramozott funkciók kimenete jelenik meg a böngészőben, egyébként pedig listázza a könyvtártartalmat.

XAMPP

A XAMMP egy ingyenesen letölthető szoftver, amely egy csomagban tartalmaz adatbázis-, web-, FTP- és levelező-kiszolgálót. Folyamatos fejlesztésének köszönhetően az újabb verziókkal a beleintegrált technológiák is mindig naprakészek. A program használatával nem szükséges ingyenes, korlátozott szolgáltatást nyújtó tárhelyet keresni, fizetni a teljes körű támogatásért vagy saját szervert felépíteni és üzemeltetni. Telepítést és konfigurálást követően saját gépünkön menedzselhetünk adatbázis- és webszervert. Jelenleg négy disztribúció áll rendelkezésre, így a Linux, a Windows, a Solaris és a MAC OS X felhasználók egyaránt élvezhetik a csomag szolgáltatásait.

A XAMPP támogatja többek közt az Apache-ot, a MySQL-t, a PHP-t és a Perl-t, a phpMyAdmin-t és a FileZilla FTP Server-t is. A beépített vezérlőpult segítségével könnyedén elindíthatjuk gépünkön a szükséges szolgáltatásokat, láthatjuk állapotukat, illetve leállíthatjuk őket.



1. ábra: A XAMMP vezérlőpanelja

Verziókezelés

Mivel a program projektmunkában készül, így elkerülhetetlen egy verziókezelő szolgáltatás használata. Fejlesztés során ugyanis mindkét fél rendelkezik a forráskód egy másolatával. Módosításkor feltétlenül szükséges a másik programozó értesítése az elvégzett módosításról, hogy a változtatások az ő kópiáján is megjelenjenek. A verziókezelő rendszer egy, olyan eszköz, mellyel ez a feladat automatizálható, így kerülve el az esetleges konfliktusok kialakulását.

Ilyen rendszer a Concurrent Versions System (CVS) és a Subverion (SVN). Mindkettő lehetővé teszi, hogy többen hozzáférjenek ugyanazokhoz a fájlokhoz. A rendszerek nyíltak és központosítottak, tehát szabadon használhatóak, illetve létezik egy webszerver, amely az aktuális és a korábbi módosításokat tárolja. Ezt hívják repository-nak. Itt tárolódik az elsődleges változat, más néven a master copy. A working copy a master másolata, amely minden fejlesztőnél megtalálható. Check out paranccsal tetszőleges számú másolat kérhető. A helyi kópia módosításai commit-tal kerülnek fel, míg naprakészen update-tel tarthatjuk. Konfliktus akkor alakulhat ki, ha többen szerkesztik ugyanazt az állományt és megpróbálják feltölteni. Néhány esetben előfordul, hogy a rendszer automatikusan képes összefésülni a

fájlokat, máskor viszont kézi beavatkozásra van szükség. A CVS hátránya, hogy nem kezeli a könyvtárak és fájlok átnevezését és mozgatását, és rosszul tűri a frissítés során megszakadó kapcsolatot. Az SVN ezeket a hibákat is javító továbbfejlesztés, emiatt döntöttünk a használata mellett. A beüzemeléshez szükség van egy kliensre és egy kiszolgálóra.

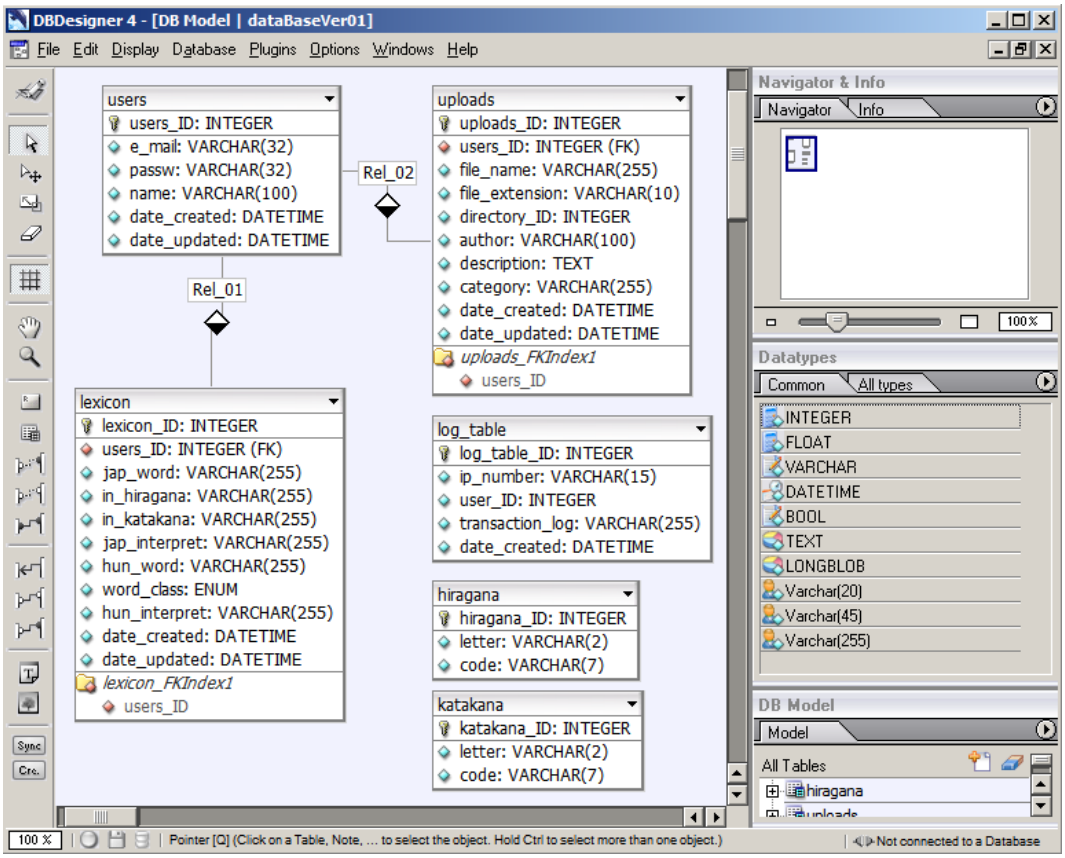
A webszerverek nagy része fizetség ellenében nyújt verziókezeléshez szükséges támogatást, de akad néhány ingyenes is, vagy akár gépünkön is üzemeltethetünk egyet. Ha üzleti célú projektet kívánunk megvalósítani érdemes ez utóbbit választani, esetleg saját szervert üzemeltetni. A megbízható, díjtalan megoldásokkal általában nyílt forrású projektek készíthetők, amely nem minden esetben jó megoldás. Ilyenek a *tigris.org* vagy a Google Projekt Hosting (*code.google.com*). Részünkről a *freesubversion.com* és az *unfuddle.com* lettek kipróbálva. Az előbbi egy egyszerű webes felületet nyújt csupán, amin felhasználókat adhatunk hozzá e-mail cím és név megadásával. Mivel fejlesztés alatt áll ezért nem túl megbízható alternatíva. Az unfuddle viszont teljes körű projektmenedzsment támogatást nyújt. Felhasználókat, projekteket és hozzájuk tartozó feladatokat írhatunk ki és dokumentálhatunk. Ez a megoldás azonban csak korlátozott lehetőséget biztosít azoknak a felhasználóknak, akik ingyenes szolgáltatást keresnek, mivel számukra korlátozott a létrehozható projektek és a hozzájuk köthető programozók száma, valamint az elérhető tárhely kapacitása. Azonban nekünk ez is megfelelőnek bizonyult, így a használat mellett tettük le a voksunkat.

A Tortoise SVN egy szabadon letölthető Windows-os kliens oldali program. Magyar nyelven elérhető 32 és 64 bites verziója is. Telepítés után tetszőleges könyvtárstruktúrát kialakítva, beállítva a repository címét, ha szükséges a felhasználói nevet és jelszót megadva elkészíthetjük, majd feltölthetjük az elsődleges változatot a kiszolgálóra. A Tortoise beépül az Explorerbe, így helyi menüből könnyedén és gyorsan elérhetővé válik.

DBDesigner

A DBDesigner egy freeware adatbázis tervező és fejlesztő eszköz. Használatával grafikusan jeleníthetjük meg tábláinkat és a közöttük lévő kapcsolatokat. Elérhető Linux és

Windows környezethez egyaránt. A szoftvert MySQL adatbázishoz fejlesztették ki és erre optimalizálták. Egyszerűen hozhatunk létre táblákat, változtathatunk a meglévő struktúrán, hozzáadhatunk vagy megszüntethetünk kapcsolatokat. Képes SQL szkriptet generálni az általunk kialakított szerkezetből.



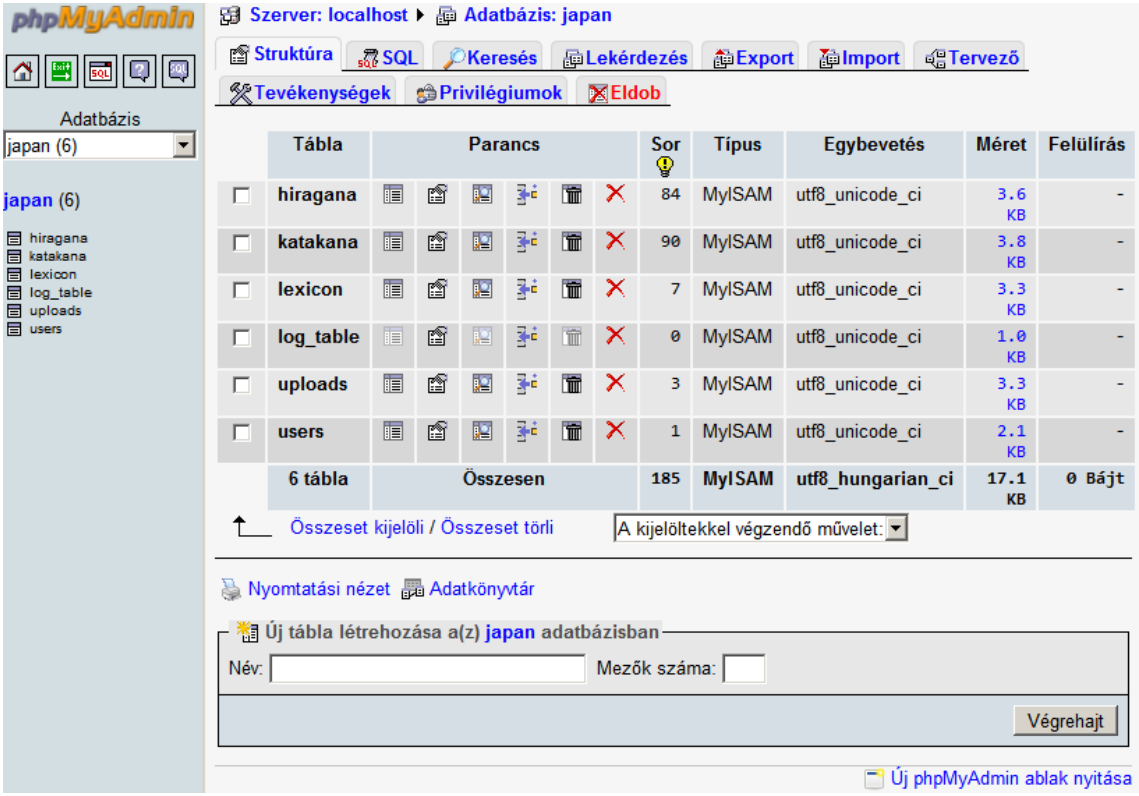
2. ábra: A DBDesigner működés közben

Adatbázis kezelők

Adatbázisunk adminisztrációjához és menedzseléséhez használhatunk webes kezelőfelületet, mint amilyen a XAMPP-ba is beépített phpMyAdmin vagy grafikus alkalmazást, amely képes elérni, és a szükséges műveleteket elvégezni tábláinkon. A rendelkezésre álló szoftverek közül a Navicat a legjobb választás.

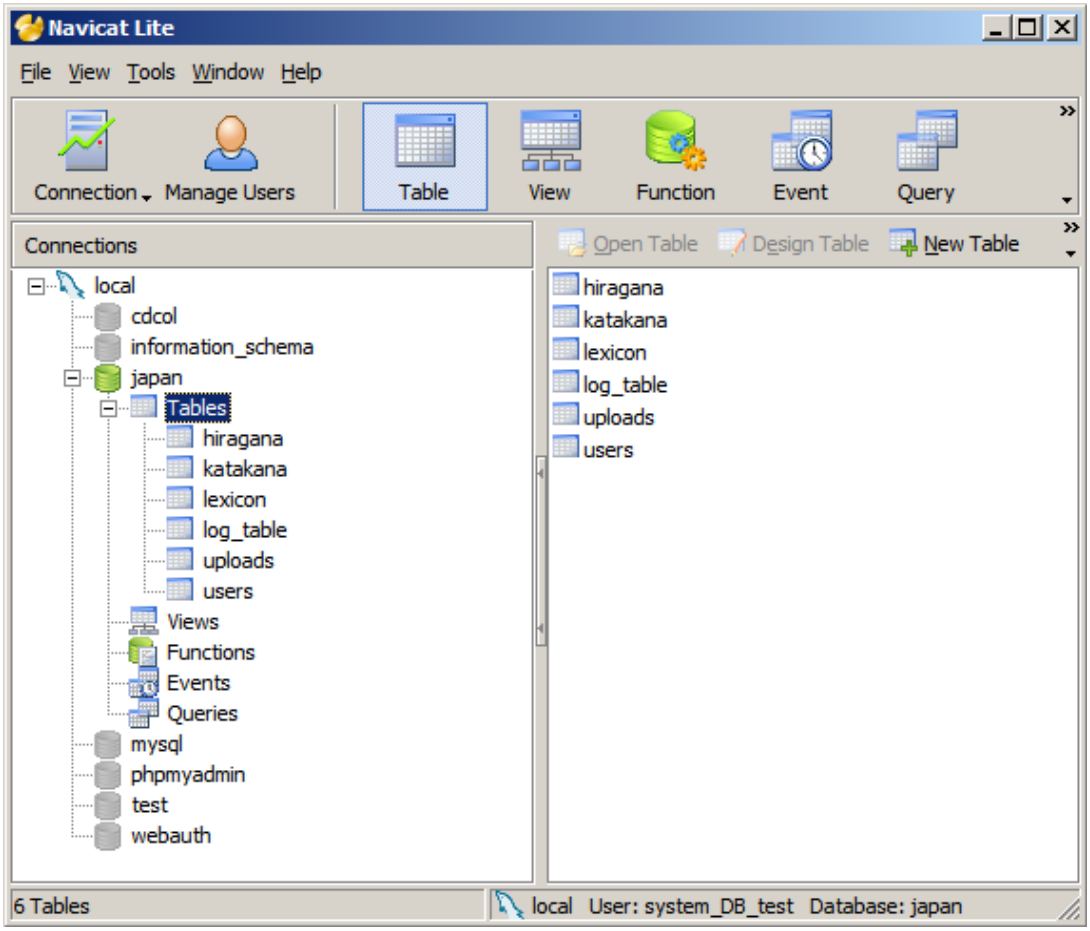
A phpMyAdmin egy open source eszköz, amit PHP-ban írtak MySQL adatbázisok

karbantartásához. Interneten keresztül lehet vele adatbázisokat létrehozni és törölni, táblákat manipulálni, valamint SQL parancsokat futtatni. Az utasítások többségét nem szükséges begépelni, mert sok funkció elérhető a phpMyAdmin felületéről, és a végrehajtást követően meg is jelenik szabványos formában.



3. ábra: A phpMyAdmin kezelőfelülete

A Navicat Premium-ból elérhető egy 30 napos demó verzió, míg a Navicat Lite egy korlátozott szolgáltatásokat nyújtó, nem kereskedelmi célokra szánt teszt változat. A program MySQL mellett Oracle és PostgreSQL adatbázisokkal is használható. Helyi vagy távoli adatbázisokhoz kapcsolódhatunk, képes SQL szkript generálásra és futtatásra. Támogatja az Unicode, az UTF–8 és egyéb karakterkészleteket is. Leszögezhetjük, hogy sok hasznos funkciójával jó alternatíva lehet a phpMyAdmin mellett.



4. ábra: A Navicat Lite kezelőfelülete

JavaScript

A JavaScript egy objektumorientált nyelv. Használatával weboldalunkat hasznos és látványos funkciókkal vértezhetjük fel. Szintaktikája hasonlít a Java programozási nyelvéhez. Hátránya, hogy különböző böngészőkkel eltérően viselkedhet, illetve letiltható, aminek következtében le sem fut. Alkalmazásával azonban informatív üzenetekkel segíthetjük a felhasználókat, vagy az adatok szerverhez küldése előtt ellenőrizhetjük azok tartalmát, kiszűrve ezzel a hibás információkat, esetleg a rosszindulatú támadásokat. Emiatt is érdemes megfontolni használatát. A JavaScript egyszerűen elhelyezhető HTML és PHP oldalakon a következő formában:

```
<script type="text/javascript">
        document.write("Hello World!");
</script>
```

Érdemesebb azonban egy külön *.js* kiterjesztésű fájlban összegyűjteni a megírt függvényeket, és a szükséges helyen behívni az állományt.

UTF-8

Az UTF-8 (Unicode Transformation Format) egy 8 bites karakterkódolási eljárás. A Unicode nemzetközi szabvány része. Rob Pike és Ken Thompson alkotta meg. Létrejöttét annak az elképzelésnek és vágynak köszönheti, hogy képesek legyünk az összes nyelvben előforduló karaktereket helyesen megjeleníteni. Internetes alkalmazások programozásánál a leggyakrabban használt kódtábla. A diplomamunkában bemutatott portál és a kiszolgáló adatbázis karakter kódolása azért is e szabvány alkalmazásával készült, mert a japán nyelvben előforduló jeleket szabványosan lehet megjeleníteni vele.

III. Program bemutatása

1. A mappaszerkezet

Annak érdekében, hogy a kód áttekinthetőbb legyen, a karbantartás pedig egyszerűbb, a következő könyvtárstruktúra került kialakításra:

```
|-- _dev
|   |-- database
|   |     |-- sql
|   |-- design
|-- project
|   |-- class
|   |     |-- database
|   |     |-- design
|   |     |-- string
|   |-- function
|   |-- settings
|-- web-project
|   |-- css
|   |     |-- images
|   |-- elements
|   |-- pages
|   |-- uploads
```

A *_dev* mappába kerültek a tervezés során elkészült dokumentációk és a specifikáció. Az adatbázishoz kapcsolható dolgokat a *database*, míg a design tervhez kötődőket a *design* könyvtárba helyeztük el. A *database/sql* mappa az adatbázist kialakító és a karbantartást segítő SQL szkripteket tartalmazza.

A *project* könyvtárban találhatók a működéshez szükséges PHP fájlok. A *function* a weblapon használt funkciókat meghatározó állományokat, a *settings* a beállításokért felelős *properties.php*-t, a *class* pedig az osztályokat megvalósító fájlokat tárolja. Az *database* az adatbázis kezelést, a *design* a megjelenítést, míg a *string* alkönyvtár a szöveges változók átalakítását végző osztálydefiníciók leírását tartalmazza.

A *web-project* tárolja a weboldalakat. A feltöltött dokumentumok is ide másolódnakl az *uploads* mappába. A *css* könyvtárba kerültek a stílusleíró fájlok, míg az *images* az oldalon megjelenő képeket foglalja magában. A *pages* tartalmazza az oldalakat, az *elements* pedig az

őket felépítő kisebb elemeket leíró állományokat.

2. Az adatbázis

Az adatbázis bemutatása

Az adatbázis létrehozásánál megoldandó feladatot rótt fel a japán írásrendszer. A szavak ugyanis háromféle módon írhatók. Tekintsük át ezeket.

- A kanji: kínaiaktól átvett írásjelek, melyek fogalmakat jelölnek. Tulajdonképpen egy képírás. Az egyszerűsítések miatt manapság már nem csak egyetlen jelentést hordoznak, mint korábban. Általában a szótöveket jelölik velük. Számuk több tízezerre tehető, amit a második világháborút követően 1800-ra korlátoztak. Később a jelkészlet számát 1945-re bővítették.

- A hiragana: egy szótag-ábécé, tehát elemei önállóan felolvasható szótagok, nem csupán hangokat jelölő betűk. A mássalhangzók közül egyedül az n-nek létezik japán átirata. A japán szavakat, a toldalékokat és a ragokat jelöli.

- A katakana: szintén egy szótagokat jelölő ábécé. Ugyanazon hangok leírására szolgál, mint a hiragana. Jelei természetesen különböznek. A katakana idegen eredetű és jövevényszavak leírására használatos.

A japán nyelv nehézsége abban rejlik szótár készítés szempontjából, hogy akár egy szón belül is találkozhatunk a három különböző írásmóddal. Mivel a szavakat le lehet írni pusztán hiraganával, illetve katakanával is, ezért a kanjik, használatát mellőzzük és a táblában csak az előbbi két leírási módnak biztosítunk helyet.

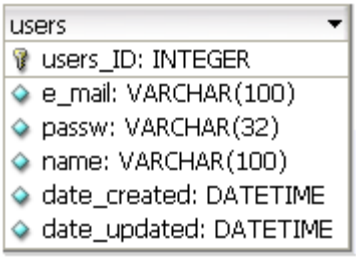
Az adatbázis tervezésénél meg kell határozni a feladathoz szükséges táblák szerkezetét. Fontos szem előtt tartani a könnyű kezelhetőséget, karbantarthatóságot és a gyorsaságot. A

szótár tábla tervezésénél két lehetőség is felvetődött. Az egyik esetben a japán és magyar szavakat külön tároljuk, és egy kapcsolótáblát alkalmazunk az N:M kapcsolat kezelésére. Ekkor három táblát kell felügyelni, és lekérdezéskor időt veszthetünk a join műveletek miatt. A másik megoldás, hogy egy táblát használunk, amely soronként tartalmazza a szavakat és a fordításukat. Ebben az esetben egyszerűbb adatokat beszúrni, lekérdezni és módosítani, viszont a rokon értelmű szavak miatt az adatbázis mérete jelentősen megnőhet. Mérlegelve az előnyöket és hátrányokat az egy táblában történő tárolás alkalmazása mellett döntöttünk. A többi tábla esetén nem ütköztünk különösebb nehézségekbe. Az adatbázis felépítését bemutató ábra a függelék I. részében található.

Minden tábla rendelkezik *PRIMARY KEY*-el, amely tíz hosszúságú előjel nélküli egész szám, és beszúráskor automatikusan növekedik eggyel. A felhasználók tábla külső kulccsal kapcsolódik a feltöltések és a szótár táblákhoz. Azoknál a tábláknál, ahol van adatbeszúrás, ott a sorok létrehozáskor időpecsétet kapnak (*date_created* oszlop), ahol lehetséges módosítás, ott az update idejét is tároljuk (*date_updated* oszlop). A létrehozott oszlopoknál használt típusok a következők:

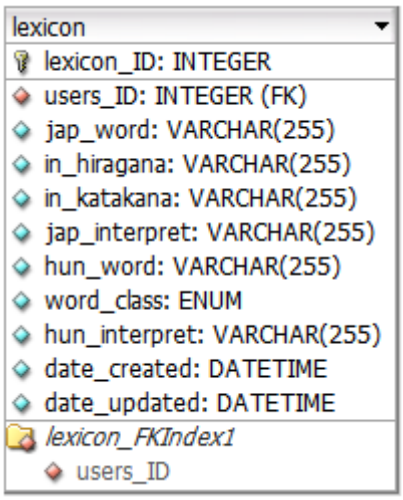
- *int*: egész szám tárolására használható. Elsődleges kulcsoknál alkalmazzuk előjel nélküli (unsigned) változatát.
- *varchar*: változó hosszúságú karaktersor, melynek mérete vagyis a leghosszabb tárolható sorozat megadható. A legtöbb szöveges adatot ebben tároljuk.
- *text*: nagyméretű szövegek kezelésére alkalmas típus. A feltöltött dokumentumok leírását tartalmazza az *uploads* táblában.
- *enum*: felsorolás típus, mely előre meghatározott halmazból kaphat értéket. A szótár táblánál a szófajok megadására használjuk.
- *datetime*: idő és dátum tárolásra alkalmas, 'év – hónap - nap óra : perc : másodperc' formátumban. A rekordok létrehozási és módosítási idejét kezeljük ezzel a típussal.

A következő táblák kerültek az adatbázisba:



5. ábra: A regisztrált felhasználókat tároló tábla szerkezete

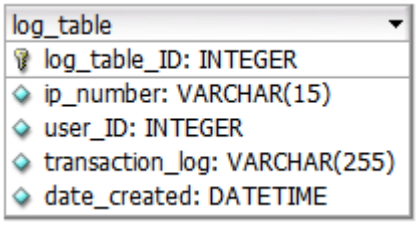
Az *e_mail* és a *passw* mező tárolja a bejelentkezéshez szükséges adatokat. Biztonsági okokból, az utóbbit a PHP-be beépített *md5* függvényével titkosítjuk. Ekkor ugyanis tetszőleges sztringből egy harminckét karakter hosszúságú kód képződik, melyből nehezen állítható vissza az eredeti szöveg. Megadható továbbá a felhasználó neve is.



6. ábra: A szótár tábla szerkezete

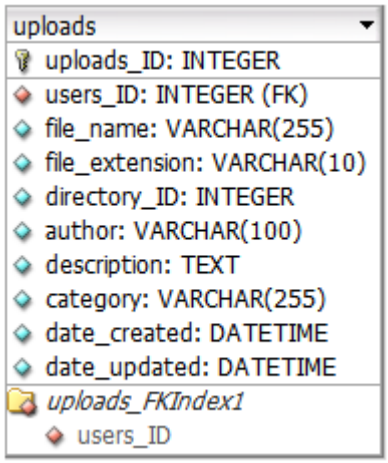
Minden sorban egy japán szót és a hozzátartozó magyar jelentést tároljuk. A szótárban a japán szavak fonetikusan találhatók meg. Lehetőség van azonban hiraganával és katakanával történő leírásra is. Ezt UTF-8 karakterkódok megadásával tehetjük meg. A *jap_interpret* és *hun_interpret* oszlopok a szavak értelmezéséhez szükséges információkat tartalmazzák. A *word_class* a szófajok megadásához szükséges felsorolásos típus. Elemei a leggyakrabban használtak rövidítései: *fn* – főnév, *hsz* – határozó szó, *ige*, *ik* – igekötő, *isz* – indulatszó, *ksz* – kötőszó, *mn* – melléknév, *nelo* – névelő, *nm*- névmás, *nu* – névutó, *szn* – számnév és *egyeb* az

összes többi tárolására.



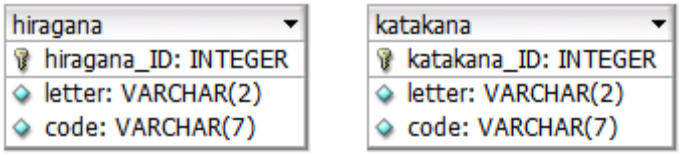
7. ábra: A naplózást tároló tábla szerkezete

A felhasználói tevékenységet külön táblába naplózzuk. Az IP cím, illetve bejelentkezett emberek esetén a *users* táblában hozzájuk tartozó azonosító is tárolódik. A végrehajtott tevékenység a *transaction_log*-ba kerül.



8. ábra: A feltöltött dokumentumokat tároló tábla szerkezete

A feltöltött állományoknak tároljuk a nevét és kiterjesztését. Opcionálisan megadható a szerző, a kategória és egy leírás. Egy mappaszerkezet is kialakítható. Ennek kezelését segíti a a *directory_ID* oszlop, mely annak a mappának az elsődleges kulcsát tartalmazza, amely az adott fájl vagy könyvtár szülője. Ha ennek az értéke nulla, akkor a feltöltött adat a gyökérbe kerül.



9. ábra: A japán írásjeleket tároló táblák szerkezete

A japán írásrendszer két elemét a hiraganát és katakanát külön táblákban tároljuk. A *letter* oszlop tárolja a szótagokat, a *code* pedig a hozzájuk tartozó UTF-8 kódokat.

Az adatbázis létrehozása

Az adatbázistáblákat létrehozó SQL parancsok a függelék II. részében tekinthetők meg. Első lépésben szükségünk van egy adatbázis kiszolgáló szerverre, amely lehet online vagy a gépünkön a XAMPP által biztosított. Ezen kell létrehoznunk adatbázisunkat a *CREATE DATABASE* paranccsal. A kapcsolódáshoz szükség lesz még egy felhasználó és a hozzá tartozó privilégiumok beállítására is. Ezt a *CREATE USER* és a *GRANT* utasításokkal tehetjük meg. A beállítási paraméterek a következők:

- adatbázis név: *japan,*
- felhasználói név: *root,*
- jelszó: *root*.

Az adatbázis, a táblák és a karakteres típusú oszlopok kódolása egyaránt UTF-8-as, amit a *DEFAULT CHARACTER SET* és a *COLLATE* kapcsolókkal állítottunk be. A tábláknál a MyISAM tárolási módot alkalmaztuk. Ennél a módszernél minden táblázathoz három állomány jön létre. A formátum, az adatok és az indexek külön fájlba kerülnek. Az *ENGINE* kulcsszóval állítható be a kívánt tárolás, A táblastruktúrát leíró SQL szkript úgy lett elkészítve, hogy újrafuttatható, mivel előbb törli a létező táblákat és csak azután hozza létre őket újra üresen.

Utolsó lépésként már csak az adatok beszúrása van hátra. Egy felhasználót kell csupán

beállítani és a japán karaktereket feltölteni *INSERT*-ekkel. A teljes struktúrát kialakító állományok a *_dev/database/sql* mappában megtalálhatók a program könyvtárában.

Az adatbázis használata

Az adatbázis kezelését a *database* alkönyvtárban lévő PHP fájlok végzik. A *config* osztály tárolja a kapcsolat kialakításához szükséges információkat publikus adattagok formájában. Az adatbázis nevét, a beállított felhasználói nevet és jelszót, valamint a szerver címét. A *database* osztály konstruktorában elvégzi a kapcsolódást és az adatbázis kiválasztást. A *class.error.php* gondoskodik a fellépő hibák kiírásáról.

Tekintsük át az adatbázis használatát egy példán keresztül. Az *index.php* állományban a

```
$database = Database::getInstance();
```

paranccsal a *$database* változóba mentjük az adatbázis egy példányát, amelyen a későbbiekben bármilyen lekérdezési, módosítási vagy törlési parancs indítását kezdeményezhetjük. Nem szükséges az adott PHP oldalon újra kapcsolódni, csak a megfelelő kérést kell összeállítani és lefuttatni. A szótárban történő keresésnél is ez a helyzet. A *$searchLanguage* változó tárolja a nyelvet, amelyben keresünk, a *$searchWord* pedig a keresett karaktersorozatot. Mindkettőre szükségünk van a *SELECT* parancs összeállításához, ugyanis előbbi határozza meg a tábla oszlopát, amelyben keresünk az utóbbi, pedig magát a szót, amit le akarunk fordítani. Nézzük meg, hogyan is jelenik meg mindez a kód szintjén:

```
if( $searchLanguage == 'magyar' ) {
     $column = "hun_word";
}
else {
     $column = "jap_word";
}
$result = mysql_query
("
     SELECT *
```

```
                FROM lexicon
                WHERE $column = '$searchWord'
        ");
```

Amint az jól látható a *$column* változóban tároljuk le a keresett nyelvnek megfelelő oszlopot, aminek értékét egy egyszerű if szerkezet segítségével határozzuk meg. A *$result* tartalmazza a lekérdezés eredményét. Ha a visszakapott sorok száma nulla, akkor nem találtuk meg a keresett szót, amit a felhasználóval közlünk a következő formában:

```
if( mysql_num_rows( $result ) == 0 ) {
     echo 'A keresett szó nem szerepel az adatbázisban.';
}
```

Ellenkező esetben legalább egy találatunk van. A kapott eredmény megjelenítéséhez programunk egyszerűen egy HTML táblázatot ír ki a képernyőre *print* fügvény alkalmazásával. Ahhoz, hogy a talált sorok mezőit egyesével tudjuk kezelni a *mysql_fetch_object* függvényt kell használnunk a következő formában:

```
while ( $rows = mysql_fetch_object( $result ) ) {
        print $rows->jap_word;
        print $rows->hun_word;
        print $rows->word_class;
}
```

Illetve lehetőség van a *mysql_fetch_array*-el is hasonló végeredmény eléréséhez:

```
while ( $rows = mysql_fetch_array( $result ) ) {
        print $rows["jap_word"];
        print $rows["hun_word"];
        print $rows["word_class"];
}
```

A while ciklussal végigmegyünk a sorokon egyesével és kiíratjuk a szükséges adatokat. A két megoldás teljesen egyenértékű. A portál az objektumos megoldást használja annyi különbséggel, hogy a táblázat kirajzolásához szükséges HTML tag-eket is tartalmazza a *print* függvény.

Ne feledkezzünk meg a munka végeztével lezárni a kapcsolatot. Programunknál ezt szintén az *index.php* állományban tesszük meg, a *database* osztály *close* metódusának meghívásával, amely a *mysql_close* függvény segítségével zárja le az adatbázist a következő módon:

```
$database->close();
```

Végül nézzük meg, hogy is néz ki a *close* metódus:

```
public function close()
{
        if ($this->resource)
        mysql_close($this->resource);
}
```

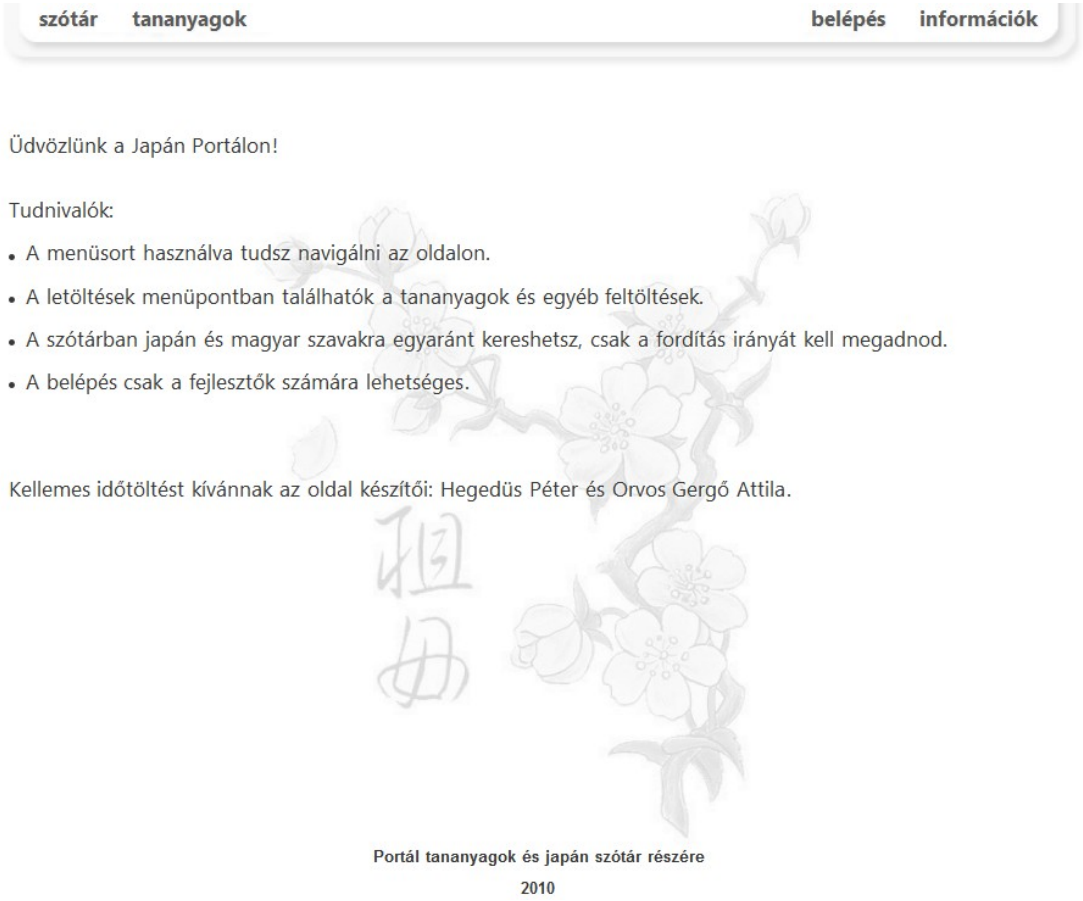
Amint az látható a *$resource* az osztály egy adattagja. Ebben tároljuk a kapcsolat azonosítót.

3. A portál bemutatása

A felhasználói oldal

A portál felülete három jól elkülöníthető részre tagolódik. A felső részen kapott helyet a menüsor, amely az elérhető oldalak navigációját segíti, alul pedig egy lábléc, amely tetszőlegesen megadott szöveget jelenít meg, esetünkben a „Portál tananyagok és japán szótár részére” feliratot. Ez a megoldás egy jól áttekinthető keretet biztosít a két egység között megjelenő tartalomnak.

A felhasználók először az információs lappal találkoznak, ugyanis ez van beállítva alapértelmezettnek. Az oldal használatával kapcsolatos tudnivalókat tartalmazza. A fenti menüből érhető el a szótár, a letölthető tananyagok, a bejelentkező képernyő és a kezdőoldal.



10. ábra: Az információs oldal

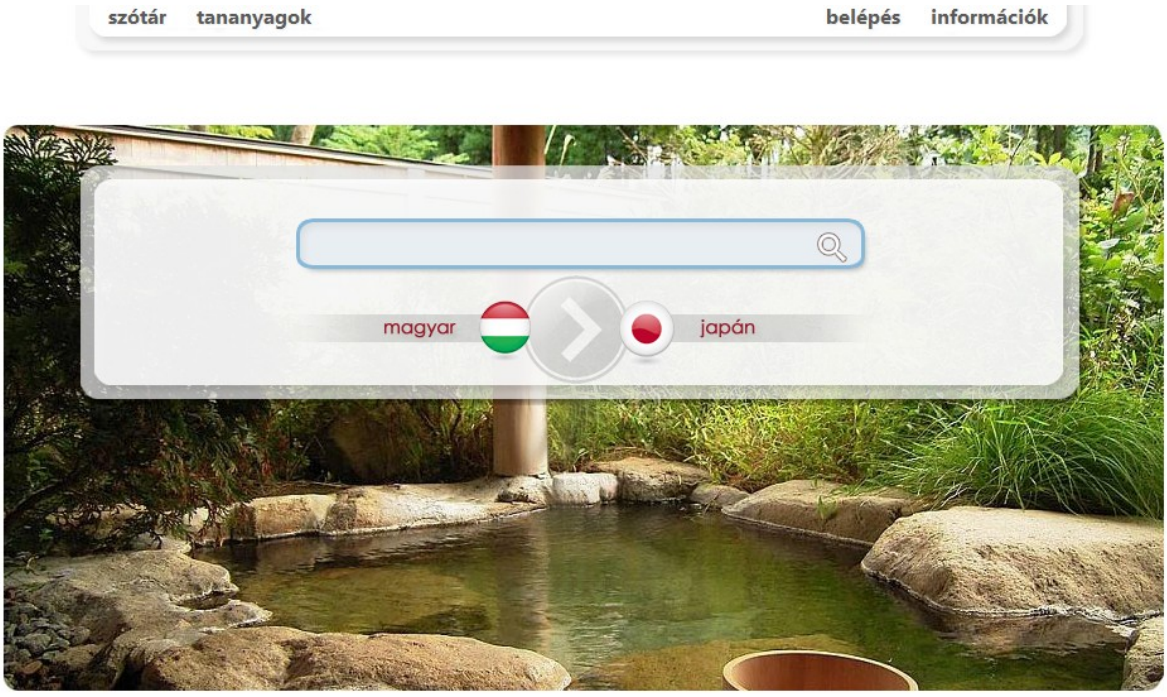
A szótár lapon a keresődoboz használatával jeleníthetjük meg a szavakat. Begépelés után kiválaszthatjuk a fordítás irányát, amit a középső nyíl mutat, amely egyben gombként is funkcionál, majd a nagyítóra kattintva vagy enter gomb megnyomását követően megtekinthetjük a találatokat.

Keresett szó: **fog**

japánul	hiraganával	katakanával	japán magyarázat	magyarul	szófaj	magyarázat
ha	は	ハ		fog	fn	
ukeru	うける	ウケル		fog	ige	
mocu	もつ	モツ		fog	ige	vmit kézben
curu	つる	ツル		fog	fn	horgászik
cukamu	つかむ	ツカム		fog	ige	megért

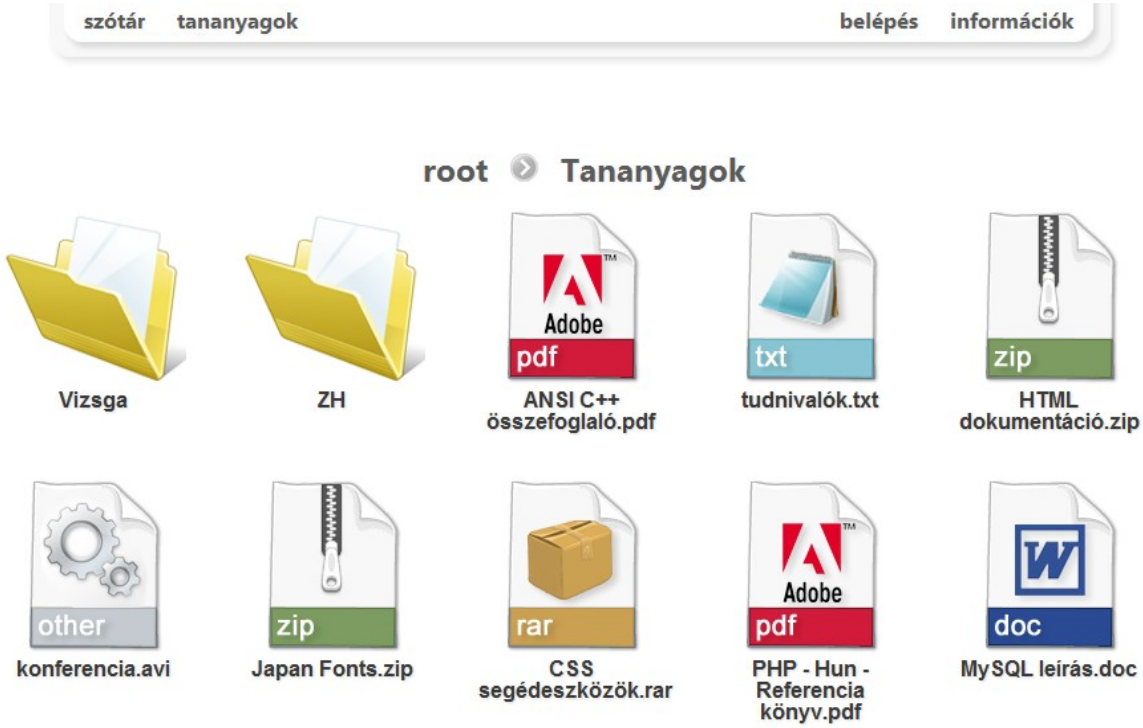
11. ábra: a keresési találatok megjelenítése

Az oldalon megjelenik a keresett szó, alatta pedig felsoroljuk a hozzá tartozó jelentéseket. Az adatokat természetesen az adatbázisból nyerjük, kiíratásuk pedig HTML táblázatban történik. Ha a szó nincs benne az adatbázisban azt jelezzük a felhasználónak.



12. ábra: A szótár főoldala

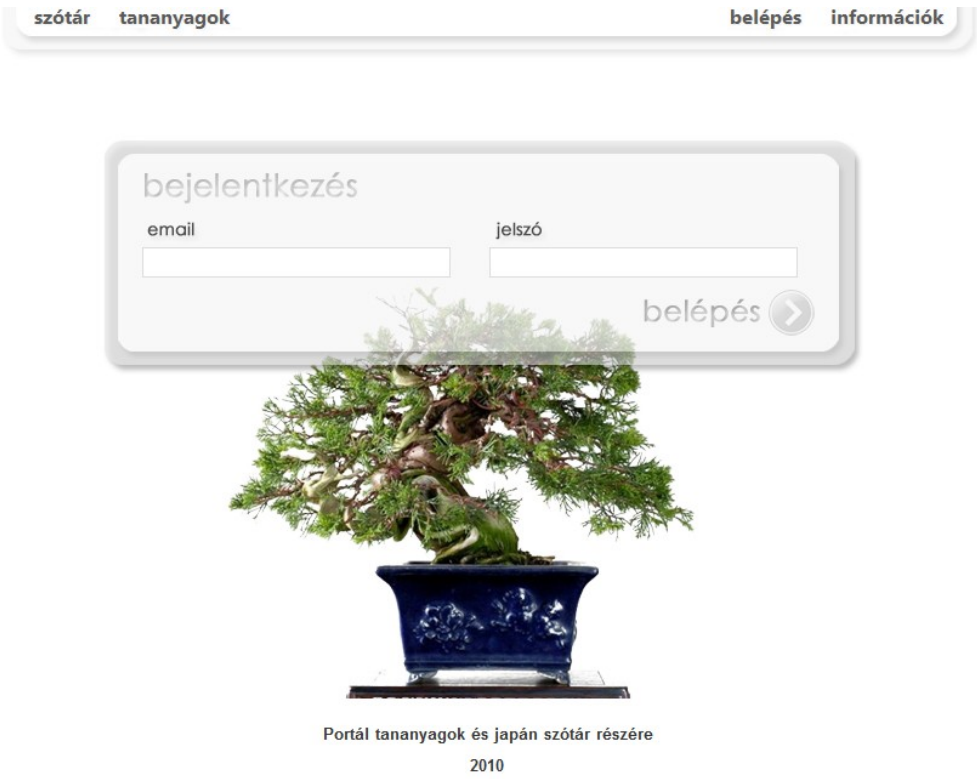
A tananyagok menüpont alatt érhetők el a feltöltött dokumentumok. Az ikonok reprezentálják a könyvtárakat és a fájlokat. Az ikon képe fájltípustól függően változik, amelyre rákattintva könyvtár esetén megjelenik a tartalom, illetve állomány esetén egy letöltés gomb. A felső részen kapott helyet egy információs sáv, mely az aktuális elérési utat mutatja. Ezen végezhető kattintással a mappák közötti navigáció. Az elérési út bármely elemére kattintva a kívánt mappába ugorhatunk. A root az alapértelmezett gyökérkönyvtár.



13. ábra: A letölthető tananyagok

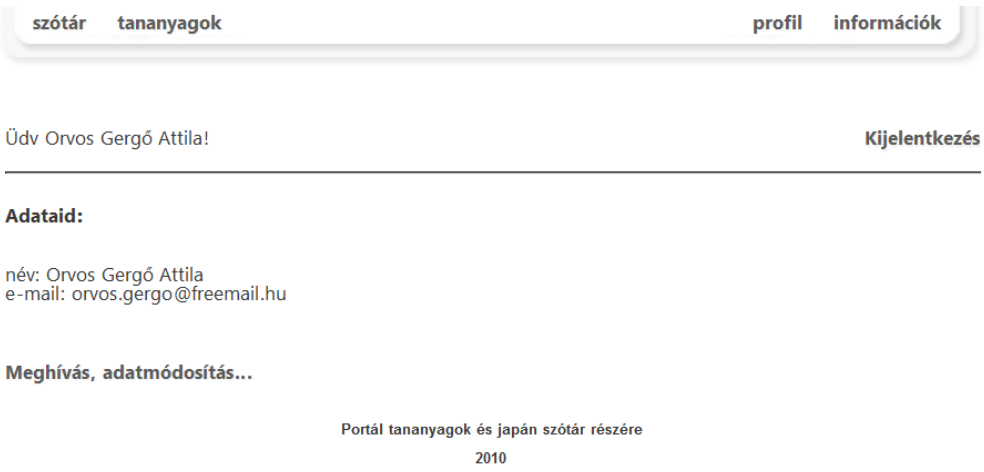
Az adminisztrátori oldal

Ahhoz, hogy elérjük az adminisztrátori funkciókat, be kell jelentkezni az e-mail cím és a jelszó megadásával. A beléptető oldalt a belépés menüpont alatt érhetjük el. A rendszer zárt körű, tehát csak egy admin hívhat meg más felhasználót az adatai beállításával. Ez természetesen nem okoz problémát, mert az összes hasznos funkció elérhető ezen jogkör nélkül is, kivéve a karbantartási feladatokat és az adatbázis módosítást. Tehát a portál elsődleges szolgáltatásaihoz, vagyis a szótár használatához és a dokumentum letöltéshez, mint azt láthattuk, nem szükséges regisztráció.



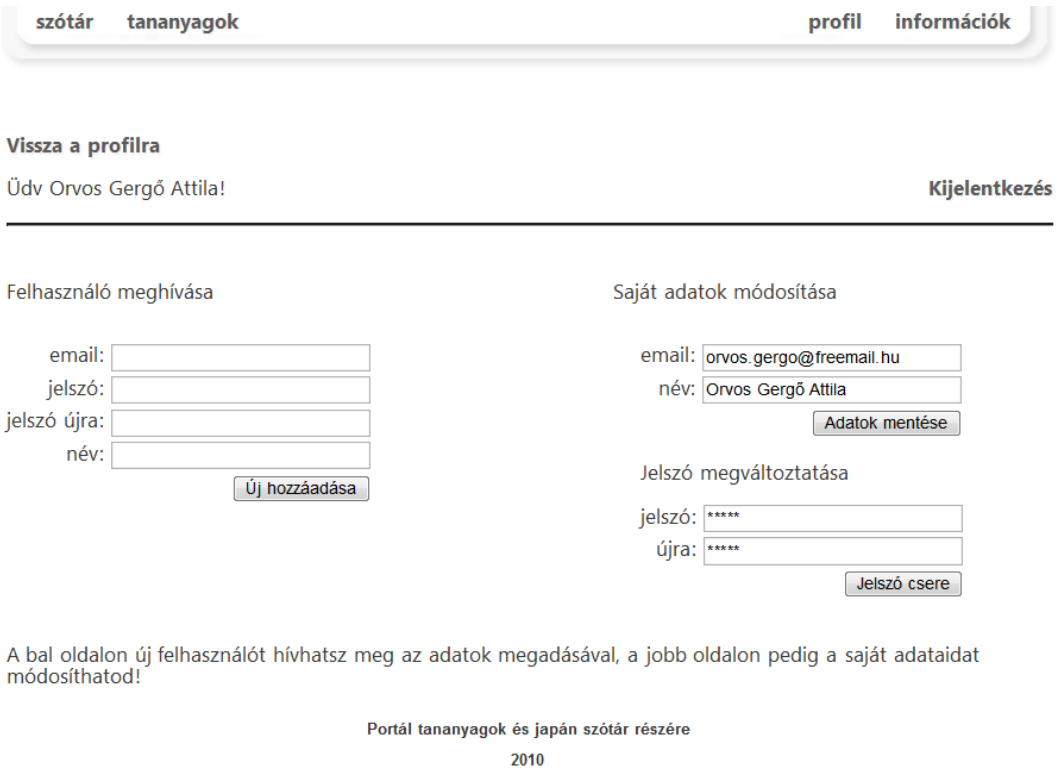
14. ábra: A login oldal

A beírt adatok ellenőrzését követően, ha azok hibásak, visszairányítjuk a felhasználót a bejelentkező képernyőre, ellenkező esetben a profil oldalra jut. Itt található a kijelentkezés és az adatmódosítás átirányítás. Utóbbira kattintva átkerülünk az admin oldalra, előbbi pedig elvégzi a kijelentkezést. Sikeres autentikációt követően a belépés menüpont profilra változik.



15. ábra: A profil oldal

Az admin oldalon vehetünk fel új adminisztrátort az e-mail cím, jelszó és név megadásával, vagy módosíthatjuk saját fiókunk adatait. A jelszavakat kétszer kell megadni az elgépeléses problémák megelőzése érdekében. Erről az oldalról is kijelentkezhetünk, vagy visszatérhetünk a profil oldalra. Az admin felület a menüből nem elérhető.

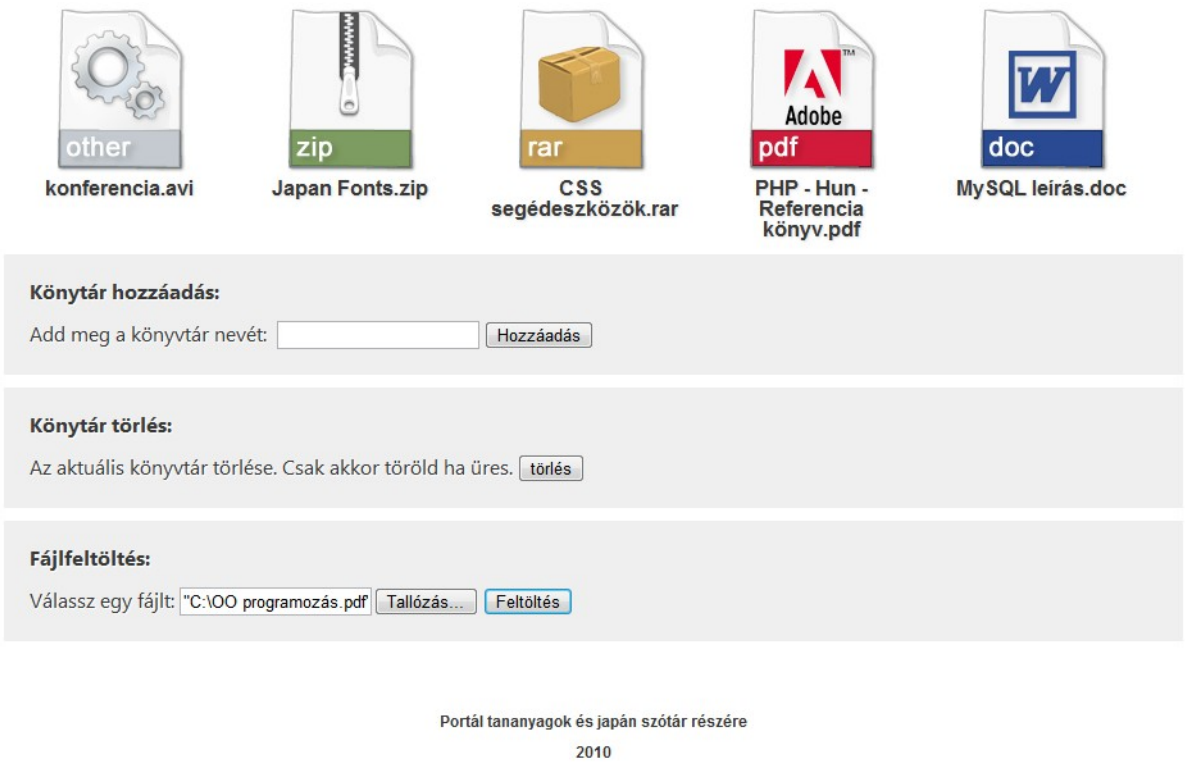


16. ábra: Az admin felület

A tananyagok és a szótár oldalon attól függően, hogy van e bejelentkezett adminisztrátor plusz funkciók jelennek meg. A megvalósításhoz több megoldás közül választhatunk. Lehetőségünk van az egyes lapok bonyolultságát növelni és minden funkciót egy állományba elhelyezni. Ez kevés nagy méretű nehezen javítható, módosítható oldalt eredményez. Rendelhetünk minden szolgáltatáshoz külön weblapot, amely már egy jobb megoldás, de mégjobb, ha kevés főoldalunkhoz a felmerülő igények alapján hívjuk meg a szükséges tartalmakat. Portálunkon is ilyen modul szerű megoldást alkalmaztunk. Szükség esetén, vagyis bejelentkezés után, formok jelennek meg a fő tartalom alatt. Ezeket külön PHP állományokban tároljuk és szükség esetén az *include_once* függvény segítségével hívjuk meg.

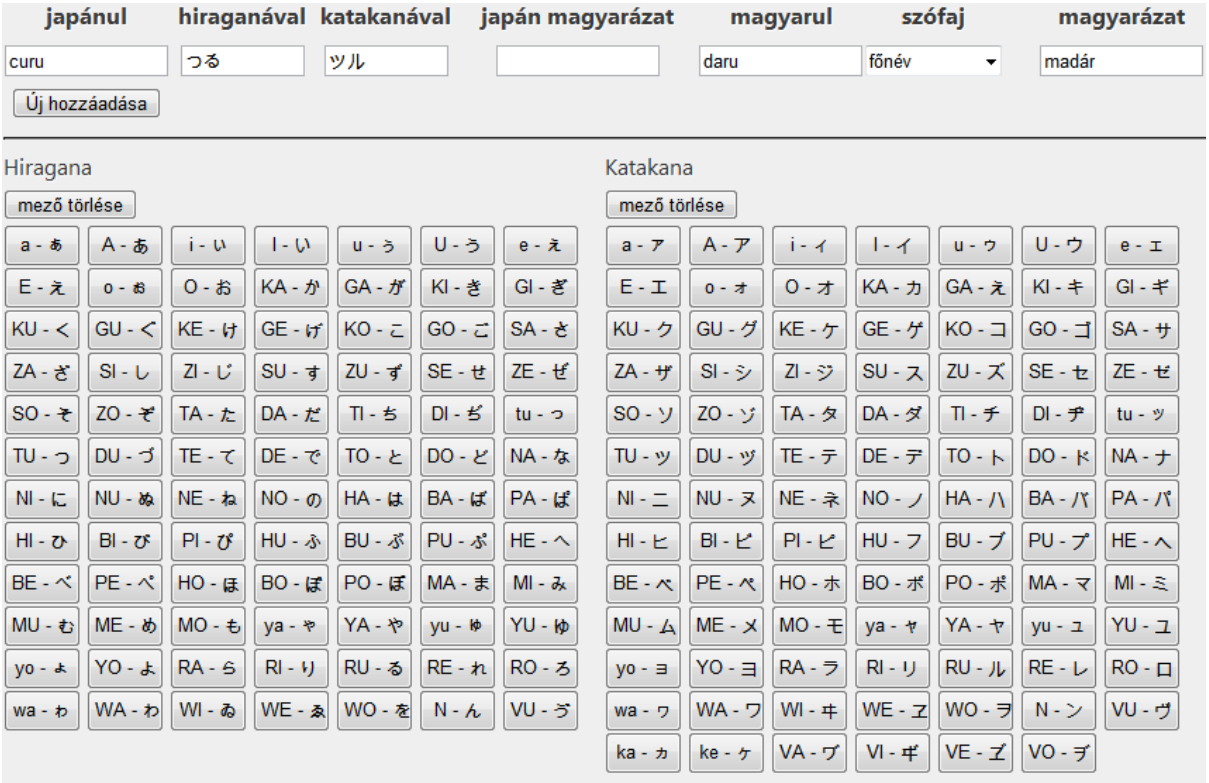
A megjelenő űrlapok használatával bővíthetjük a szótárt illetve feltölthetünk új dokumentumokat is.

Először tekintsük át a tananyagok adminisztrációs felületét. Látható, hogy minden ugyanolyan, kivéve az alul megjelenő űrlapokat. Felül új könyvtárt hozhatunk létre a nevének megadásával. Az új mappa mindig az aktuális szinten fog létrejönni. Valójában ez egy logikai könyvtárstruktúra lesz, mivel a merevlemezen nem jön létre a hierarchia, csak az adatbázisban. Azonban minden állományról és mappáról el tudjuk dönteni az adatbázis megfelelő oszlopa alapján, hogy hova kerüljenek listázáskor. A középső részen dobhatjuk el az aktuális könyvtárt. Ez a funkció fájlok adatbázisbeli adatait is képes kitörölni, aminek következtében nem listázódnak. Azonban a mappákkal ellentétben, a feltöltött dokumentumok fizikailag is megjelennek a merevlemezen, így a helyfelszabadítást magunknak kell elvégeznünk. Az alsó részen a tallózás gomb megnyomását követően kiválaszthatjuk a feltölteni kívánt fájlt. Majd a feltöltésre kattintva elküldjük a szervernek.



17. ábra: A tananyagok admin felülete

A szótár esetében is egy űrlappal végezhetjük el a bővítést. A fonetikus leírás, a szükséges magyarázatok és jelentés megadása egyszerűen begépelhető. A szófaj beállítását kiválasztómenü használatával oldottuk meg. Lehetőség van a hiraganával és a katakanával történő felírásra is. Ezt két darab képernyő-billentyűzet kialakításával oldottuk meg, így a két írásrendszer nem keveredik össze. A gombokon megjelenik a szótag, latin írásmóddal, valamint a japán megfelelője is, a könnyebb beazonosíthatóság érdekében. Egérkattintásra a gombnak megfelelő jel a megfelelő oszlopba kerül. Lehetőség van továbbá az elgépelt tartalom törlésére is, amelyre a mező törlése gomb használható.



18. ábra: A szótár admin felülete

A folyamatok bemutatása kódolási szinten

Az *index.php* állományban végezzük az oldalak kezelését. Attól függően, hogy milyen információkat adunk át az URL-nek vagy formok action részének, töltődik be a kívánt

weboldal. Nem létező hivatkozás esetén az alapértelmezett tartalom hívódik meg. Portálunk esetében ez az információs lap. A kezelést switch case szerkezettel végezzük:

```
switch ($page) {
   case 'szotar':
        $currentPage = './pages/search_main.php';
        break;
        …
   default:
        $currentPage = './pages/info.php';
        break;
}
```

Az adminisztrátori bejelentkezést a PHP-be beépített *session* függvény segítségével valósítjuk meg. A folyamatot az *index.php* ben lévő *session_start* parancs indítja. A sikeres autentikációt követően a *$_SESSION['login']* globális változó a *true*, míg a *$_SESSION['name']* a felhasználó adatbázisban tárolt nevét, a *$_SESSION['id']* pedig a táblabeli sorának elsődleges kulcsát kapja értékül. A kulcs tárolása a táblákat módosító műveletek miatt kell, és így nyomon követhetővé válnak a tevékenységek. A *$_SESSION['login']* ellenőrzésével tudjuk megállapítani, hogy adminisztrációs vagy normál nézetet jelenítsünk meg. A következő kódrészlet biztosítja, hogy ugyanazon az oldalon jeleníthessünk meg, vagy rejtsünk el plusz információkat, funkciókat a jogosultságnak megfelelően:

```
if( isset($_SESSION['login']) && $_SESSION['login'] === true )
     {
          Tevékenységek megadása...
     }
```

Vagyis ha létezik *login* érték és a *true*-t tárolja, akkor végrehajtja a blokkban megadott utasításokat. Ezt a módszert használjuk az összes olyan helyen, ahol lehetséges adminisztrátori interakció. A *$_SESSION['name']* változót személyes üdvözlésre és módosító formnál alapértelmezett érték megadására használjuk. Kijelentkezéskor a *logout.php* oldalon található *session_destroy* függvény hívódik meg, amely megsemmisíti a munkafolyamathoz rendelt összes adatot. Vagyis a fenti if hamis értékkel fog visszatérni és nem hajtódnak végre a

parancsok, ami a normál felhasználóknak szánt felületet eredményezi.

Végezetül nézzük meg hogyan is készíthetünk japán nyelvű billentyűzetet. Maga a kód a *new_lexicon.php* állományban az *elements* mappában található. Ezt hívjuk meg a következő módon a *search_main.php* fájlban:

```
include_once './elements/new_lexicon.php';
```

A *new_lexicon.php* tulajdonképpen egy űrlapot és a feldolgozásához, illetve a megadott adatok alapján végbemenő, adatbázis módosítást végrehajtó utasítás blokkot tartalmaz. A billentyűzet dinamikusan épül fel az adatbázisban tárolt hiraganák és katakanák felhasználásával. A gombok egyszerű HTML gombok, amelyeket egy táblázatmezőben jelenítünk meg. A feliratokat közbeékelt PHP kóddal nyerjük ki az adatbázisból az alábbi módon:

```
…
<td>
<?php
$result = mysql_query("SELECT * FROM hiragana");
while($row = mysql_fetch_object( $result ) ) {
     $code = $row->code;
     $letter= $row->letter;
     $val = $letter." - ".$code;
?>
<input type = "button" name = "gomb" value="<?= $val ?>"
     OnClick = "hiragana.value += '<?= $code ?>'">
<?php } ?>
</td>
…
```

Az *OnClick* attribútum fogja elvégezni Javascript alkalmazásával a szövegdobozba írást, amelyre nevével hivatkozunk.

IV. Összefoglalás

Amint azt tapasztalhattuk, egy dinamikus portál fejlesztése legyen az bonyolult vagy egyszerű felépítésű, igen sok odafigyelést, tervező munkát és technológiai ismereteket igényel. Erre diplomamunkám készítése közben többször is felfigyeltem. Ahhoz, hogy kézzelfogható eredményeket érjünk el, feltétlenül szerepet játszik az átgondolt célkitűzés.

Az adatbázis megtervezése sok időt és energiát igényel, mivel az alapját képezi egy dinamikus portálnak. Még egy ilyen kis projektnél is, mint amilyen a miénk volt az elején ráfordított erőforrások sokszorosan megtérülnek azáltal, hogy utólagosan nem kell módosítással, újratervezéssel tölteni az időt.

A feladat elvégzése során sikerült bepillantást nyernem a projektmunkában folytatott fejlesztés viszontagságaiba és előnyeibe egyaránt. Mivel az üzleti életben túlnyomó részt egy egész csapat dolgozik egy-egy szoftver megalkotásán, így a megszerzett tapasztalataim mindenképp előnyömre válnak majd.

A diplomamunka eredményeként létrejött egy egyszerű felépítésű dinamikus portál, mely magában rejti a továbbfejlesztés lehetőségét. Nem minden tervezett funkció lett ugyan leimplementálva, és a megvalósítottakon is lehet javítani, ennek ellenére a meglévő legfontosabb szolgáltatások működőképesek. A mai internet központú világban, ahol minden és mindenki elérhető, sőt jelentős szellemi, kulturális és anyagi hasznot is képes produkálni, semmiképpen nem felesleges egy online alapokon álló program elkészítése. Természetesen szoftverünk nem tökéletes, de nem is ez volt az elsődleges szempont, hanem az, hogy megpróbáljuk bemutatni, mivel is jár egy ilyen jellegű honlap és a hozzá tartozó adatbázis elkészítése. Úgy gondolom ezt a célt sikerült elérnünk.

A fejlesztés során felmerült néhány probléma, melyek magukban hordozzák a további fejlesztési módokat. Ilyen például az SQL injection támadásokkal szembeni védelem. Ez a fajta rosszindulatú tevékenység a beviteli mezőkön keresztül történő SQL parancsok futtatására irányul. Használatával ki lehet kerülni a validációs folyamatot, esetleg olyan kárt

képes okozni, amely a rendszer összeomlásához vezet. Kivédése validációs folyamat közbeiktatását igényli, azaz mindig ellenőriznünk kell a felhasználók által megadott adatokat. A portálon két helyen, a szótárnál és a belépésnél van lehetőség külső irányból érkező támadásra. Mivel a szótár karakterekké alakítja a bemenetet és kiszedi azokat amelyek nem betűk, így itt nem lehetséges élni ezzel a támadási móddal. Bejelentkezéskor az e-mail cím formáját ellenőrizzük, illetve azt is, hogy nem üres-e ez a mező. A továbblépéshez azonban elengedhetetlen az adminisztrátori oldalról érkező kérések ellenőrzése is.

Jó fejlődési irány lehet egy online közösség kialakítása. Ez a japán nyelv iránt érdeklődők számára, a nyelvtani szabályok elsajátításához szükséges tananyagok elhelyezésével kezdhető meg. Hírlevél szolgáltatás, blog, fórum vagy tanulást segítő, weben kitölthető feladatlapok integrálása a portálba tovább növelheti a látogatottságot. A japán kulturális és művészeti kincsek bemutatásával lehetne kiterjeszteni a funkcionalitást, esetleg egy webshop motor megírásával, a keleti tárgyak iránt fogékony emberek érdeklődését felkeltve, egyúttal üzleti alapokra helyezni a projektet.

A lehetőségek, mint látjuk végtelenek. Ragadjuk meg őket!

V. Irodalomjegyzék

Nyomtatott szakirodalom

[1] Matt Zandstra: Tanuljuk meg a PHP4 használatát 24 óra alatt

[2] Matt Zandstra: Tanuljuk meg a PHP5 használatát 24 óra alatt

[3] Peter Moulding: PHP Haladóknak – Fekete Könyv

[4] Steve Suehring: MySQL Bible

[5] Sági Gábor: Webes adatbázis-kezelés MySQL és PHP használatával

[6] Tanaka Akio: Bevezetés a japán szókészletbe

Elektronikus szakirodalom

[7] Stig Sæther Bakken, Egon Schmid: PHP kézikönyv

[8] Dr. L. Nagy Éva: SQL röviden 2007.
http://delfin.unideb.hu/~lnagyeva/SQL/

[9] Csernai Csaba: Verziókövető rendszerek 2008.
http://ganymedes.lib.unideb.hu:8080/dea/bitstream/2437/5410/1/Szakdolgozat.pdf

[10] http://php.net/docs.php

[11] http://en.wikipedia.org/wiki/PHP

[12] http://en.wikipedia.org/wiki/Mysql

[13] http://www.w3schools.com/js/default.asp

[14] Csernai Csaba: Verziókövető rendszerek 2008.

[15] http://www.cs.bme.hu/~egmont/utf8/

[16] 1http://www.inf.bme.hu/ooret/2000osz/ENV-01/cvs/cvs.html

[17] 1http://japanpelcek.freeblog.hu/files/hiragana.html

[18] http://japanpelcek.freeblog.hu/files/katakana.html

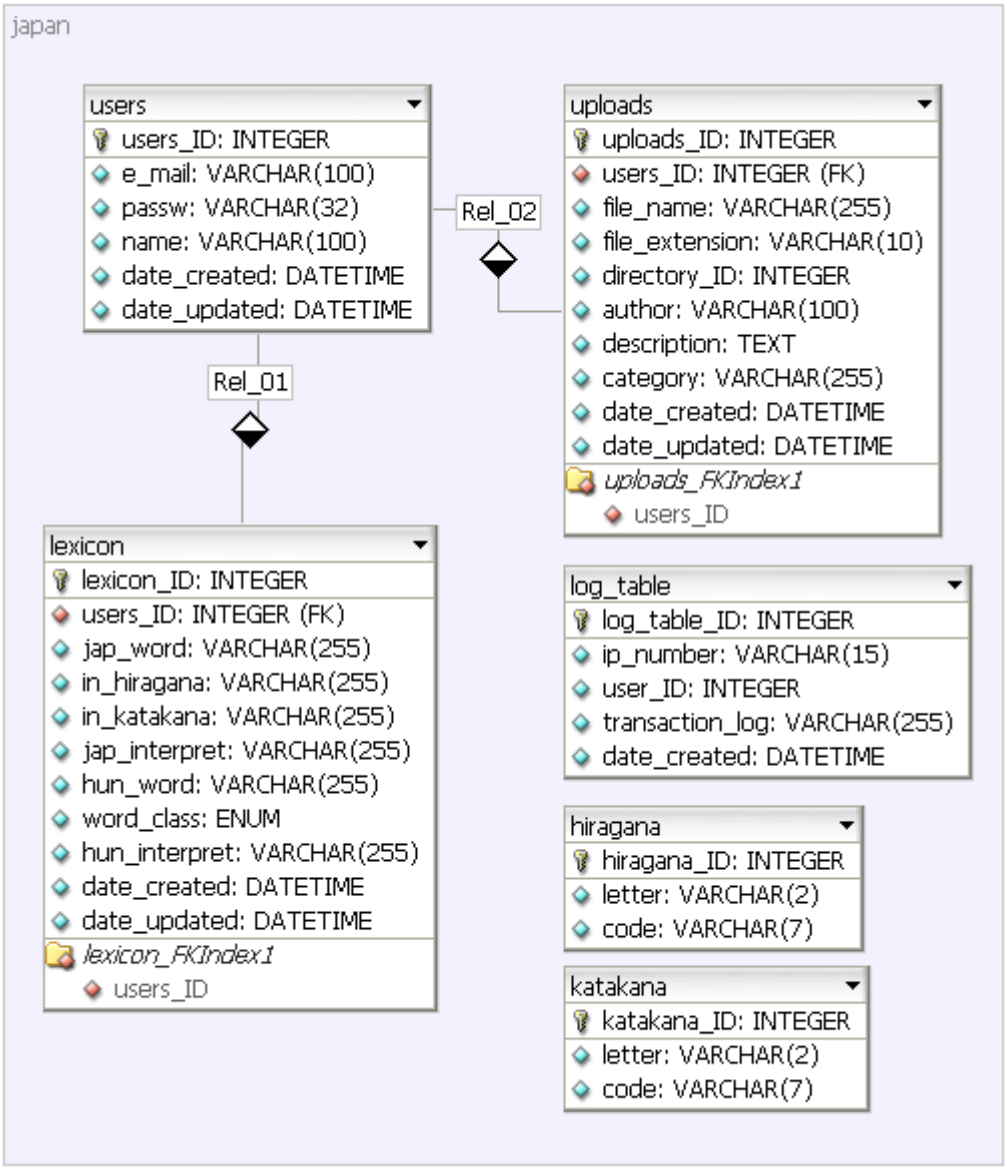
[19] http://japanpelcek.freeblog.hu/files/kanji.html

[20] http://news.netcraft.com/

[21] http://www.port80software.com/surveys/top1000webservers/

VI. Függelék

I. A z adatbázisterv:



II. A z adatbázist létrehozó SQL utasítások:

1. Az adatbázis

```
DROP DATABASE IF EXISTS `japan`;
```

```
CREATE DATABASE `japan`
      DEFAULT CHARACTER SET utf8 COLLATE  utf8_unicode_ci;
```

2. A felhasználó

```
CREATE USER 'root'@'localhost' IDENTIFIED BY 'root';

GRANT ALL PRIVILEGES ON * . * TO 'root'@'localhost'
      IDENTIFIED BY 'root'
      WITH GRANT OPTION
            MAX_QUERIES_PER_HOUR 0
            MAX_CONNECTIONS_PER_HOUR 0
            MAX_UPDATES_PER_HOUR 0
            MAX_USER_CONNECTIONS 0 ;
```

3. A táblák

- A szótár tábla

```
DROP TABLE IF EXISTS `lexicon`;

CREATE TABLE `lexicon`
(
`lexicon_ID` int(10) unsigned NOT NULL auto_increment,
`users_ID` int(10) unsigned NOT NULL,
`jap_word` varchar(255) collate utf8_unicode_ci NOT NULL,
`in_hiragana` varchar(255) collate utf8_unicode_ci default NULL,
`in_katakana` varchar(255) collate utf8_unicode_ci default NULL,
`jap_interpret` varchar(255) collate utf8_unicode_ci default NULL,
`hun_word` varchar(255) collate utf8_unicode_ci NOT NULL,
`word_class` enum
      (
      'fn','hsz','ige','ik','isz','ksz',
      'mn','nelo','nm','nu','szn','egyeb'
      ) collate utf8_unicode_ci default 'egyeb',
`hun_interpret` varchar(255) collate utf8_unicode_ci default NULL,
`date_created` datetime default NULL,
`date_updated` datetime default NULL,
PRIMARY KEY (`lexicon_ID`),
KEY `lexicon_FKIndex1` (`users_ID`)
)
ENGINE=MyISAM AUTO_INCREMENT=1
DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

- A regisztrált felhasználókat tároló tábla

```
DROP TABLE IF EXISTS `users`;

CREATE TABLE `users`
(
```

```
`users_ID` int(10) unsigned NOT NULL auto_increment,
`e_mail` varchar(100) collate utf8_unicode_ci NOT NULL,
`passw` varchar(32) collate utf8_unicode_ci NOT NULL,
`name` varchar(100) collate utf8_unicode_ci NOT NULL,
`date_created` datetime default NULL,
`date_updated` datetime default NULL,
PRIMARY KEY  (`users_ID`)
)
ENGINE=MyISAM AUTO_INCREMENT=1
DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

- A naplózást tároló tábla

```
DROP TABLE IF EXISTS `log_table`;

CREATE TABLE `log_table`
(
`log_table_ID` int(10) unsigned NOT NULL auto_increment,
`ip_number` varchar(15) collate utf8_unicode_ci NOT NULL,
`users_ID` int(10) unsigned default NULL,
`transaction_log` varchar(255) collate utf8_unicode_ci NOT NULL,
`date_created` datetime default NULL,
PRIMARY KEY (`log_table_ID`)
)
ENGINE=MyISAM AUTO_INCREMENT=1
DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

- A feltöltött dokumentumokat tároló tábla

```
DROP TABLE IF EXISTS `uploads`;

CREATE TABLE `uploads`
(
`uploads_ID` int(10) unsigned NOT NULL auto_increment,
`users_ID` int(10) unsigned NOT NULL,
`file_name` varchar(255) collate utf8_unicode_ci NOT NULL,
`file_extension` varchar(10) collate utf8_unicode_ci NOT NULL,
`directory_ID` int(10) unsigned NOT NULL,
`author` varchar(100) collate utf8_unicode_ci default NULL,
`description` text collate utf8_unicode_ci default NULL,
`category` varchar(255) collate utf8_unicode_ci default NULL,
`date_created` datetime default NULL,
`date_updated` datetime default NULL,
PRIMARY KEY  (`uploads_ID`),
KEY `uploads_FKIndex1` (`users_ID`)
)
ENGINE=MyISAM AUTO_INCREMENT=1
DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

- A hiragana ábécé jeleit tároló tábla

```
DROP TABLE IF EXISTS `hiragana`;

CREATE TABLE `hiragana`
(
`hiragana_ID` int(10) unsigned NOT NULL auto_increment,
`letter` varchar(2) collate utf8_unicode_ci NOT NULL,
`code` varchar(7) collate utf8_unicode_ci NOT NULL,
PRIMARY KEY (`hiragana_ID`)
)
ENGINE=MyISAM AUTO_INCREMENT=1
DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

- A katakana ábécé jeleit tároló tábla

```
DROP TABLE IF EXISTS `katakana`;

CREATE TABLE `katakana`
(
`katakana_ID` int(10) unsigned NOT NULL auto_increment,
`letter` varchar(2) collate utf8_unicode_ci NOT NULL,
`code` varchar(7) collate utf8_unicode_ci NOT NULL,
PRIMARY KEY (`hiragana_ID`)
)
ENGINE=MyISAM AUTO_INCREMENT=1
DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

- A táblák tartalmát törlő parancsok

```
TRUNCATE `lexicon`;
TRUNCATE `log_table`;
TRUNCATE `uploads`;
TRUNCATE `users`;
TRUNCATE `hiragana`;
TRUNCATE `katakana`;
```

VII. Köszönetnyilvánítás

Köszönetet mondok mindazoknak,
akik segítségemre voltak a nehéz időkben,
és hozzájárultak célom eléréséhez.

Köszönet illeti Dr. Horváth Géza témavezetőmet,
aki lehetőséget biztosított diplomamunkám megírásához,
és tanácsaival segítette elkészülését.

Köszönetemet fejezem ki Hegedüs Péternek,
aki fejlesztési tapasztalatával nagymértékben hozzájárult
a projekt megvalósításához.