Debreceni Egyetem
Informatikai Kar

Webes felülettel ellátott szakdolgozat-kezelő rendszer készítése

Témavezető:
Pánovics János
egyetemi tanársegéd

Készítette:
Szathmáry János
mérnök informatikus

Debrecen
2010

Tartalomjegyzék

Ábrajegyzék

1. Bevezetés

A választott feladatom egy olyan rendszer létrehozása volt, amely segítségével a szakdolgozatot készítő hallgatókat lehet egyszerűen és könnyen adatbázisba rendezni. Mivel jelenleg az Informatikai Karon semmilyen lehetőség nincs arra, hogy a hallgatók szakdolgozattal kapcsolatos adatait valamilyen elektronikus formában tároljuk, ezért létrehoztam egy szakdolgozat-kezelő rendszert, amely PHP, MySQL és JavaScript alapokon nyugszik. A programon belül lehetőség van új hallgató felvételére, a rendszerben lévő hallgatók listázására, valamint adataik módosítására, akár egyenként, akár tömegesen, és természetesen törölhetjük is őket. Többek között ezeket az opciókat fogom bemutatni a dolgozatom további részében. Emellett bemutatom, hogy hogyan épül fel a szerver, amin az egész rendszer fut, illetve hogy milyen buktatókba ütköztem a rendszer tervezése és elkészítése során, valamint hogy mikre érdemes odafigyelni egy ilyen program készítésénél.

A megvalósításhoz – mint már említettem – a PHP scriptnyelvet használtam. Annak, hogy ezt a nyelvet választottam, több oka is van. Az első és legfontosabb, hogy szerettem volna megismerni ezt a programnyelvet is. A legfőbb célom tehát némi tapasztalatszerzés volt, ha majd befejezem az egyetemi tanulmányaimat, akkor akár ezen ismeretekkel is el tudjak helyezkedni. A másik ok, amiért a fentebb említett megoldást preferáltam a többi nyelvvel szemben, hogy a PHP legfőbb felhasználási területe a dinamikus weboldalak készítése. A weboldalaknak sok hasznos tulajdonsága van, de jelen esetben a legfőbb szempont az volt, hogy a rendszert bárhonnan el lehessen érni, és bárki, aki rendelkezik egy JavaScriptet támogató böngészővel, valamint a megfelelő azonosító-jelszó párossal, az hozzáférhessen a felülethez. Természetesen a kliensgépnek valamilyen módon csatlakoznia kell a szerverhez, ez lehet belső hálózaton, vagy akár az interneten keresztül is. Így lényegében nem kell semmilyen programot telepíteni a célgépre, és nem kell előzetes konfigurációt sem elvégezni ahhoz, hogy hozzáférhessünk a hallgatókat tároló rendszerhez. Ez nagyon hasznos, mert ahogyan én észrevettem, az egyetemen a legtöbb program telepítéséhez, illetve ahhoz, hogy egy adott program kommunikálhasson a hálózat tetszőleges portjain, rendszergazdai jogosultságok kellenek. Ezt ezzel a webes megoldással teljes egészében kiküszöböljük. Annak köszönhetően, hogy egy webes rendszert készítettem, lehetőségem nyílt betekintést nyerni a Linux szerverek konfigurációjába, mivel szükség volt egy kiszolgálóra.

Ennek nagyon örültem, mert a másik terület ami engem igazán érdekel az informatikán belül, az a rendszergazdai feladatok ellátása.

2. A felhasznált nyelvek bemutatása

2.1 PHP (szerver oldali programnyelv)

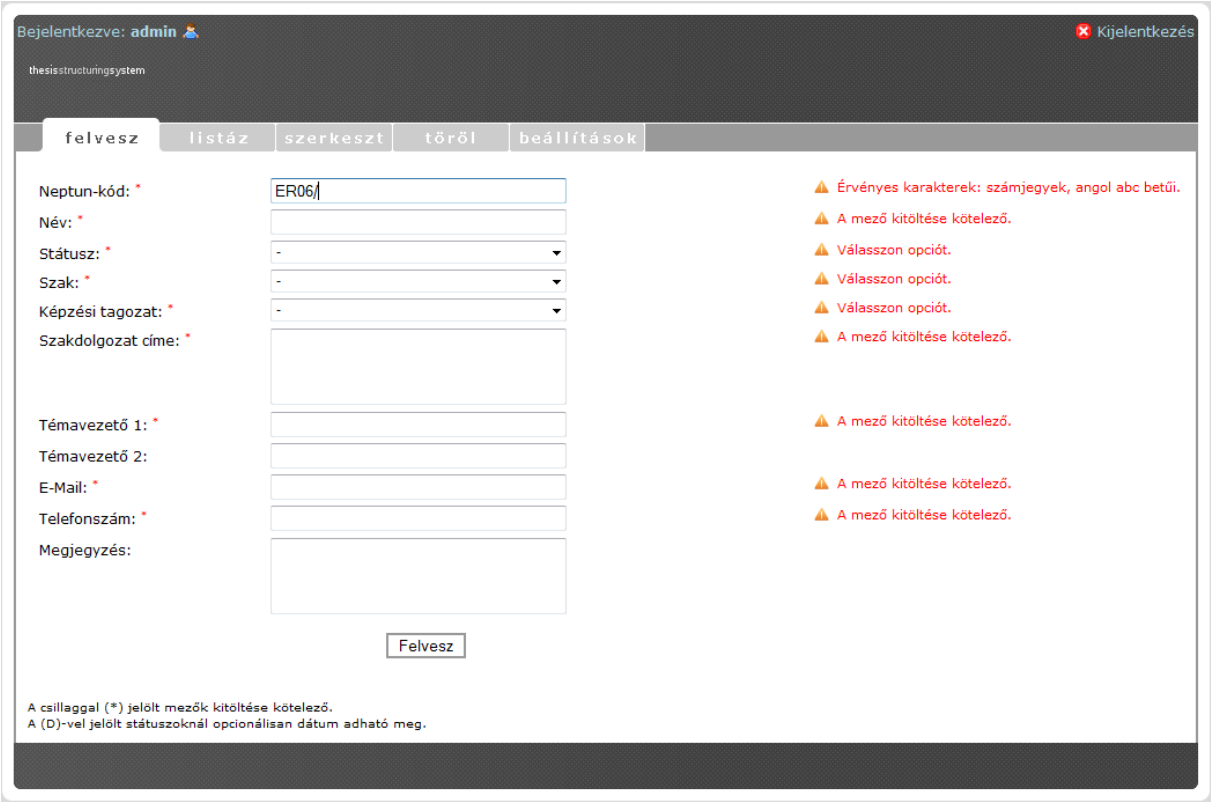
Ebben a részben szeretném bemutatni egy kicsit részletesebben a PHP nyelvet. A PHP mellett természetesen még léteznek más, hasonló feladatokat ellátó nyelvek is, mint például a Ruby on Rails mely a Ruby programozási nyelvre épülő, nyílt forrású webalkalmazás-keretrendszer. Az interneten nagyon sok keretrendszer (framework) elérhető a PHP-hoz, amelyek már kész megoldásokat nyújtanak egyes problémákra, mely nagyban megkönnyítheti a programozó dolgát, ez az egyik nagyon hasznos tulajdonsága a keretrendszereknek. A másik pedig, hogy szinte mindegyik követ egy bizonyos konvenciót, mely nem más, mint az MNV, vagyis Modell-Nézet-Vezérlő (Model-View-Controller – MVC) minta. Ez egy szoftvermérnöki munkában használt szerkezeti minta, és nem csak ebben a programnyelvben használható. Amolyan programozási struktúra, ami azt szolgálja, hogy szétválasszuk a programunkat, ezáltal is megkönnyítve és „tisztábbá” téve a kódot. Arról, hogy ezeket én hogyan valósítottam meg és milyen programozási mintákat követtem, a dolgozatom további részében fogok kitérni. A PHP (PHP: Hypertext Preprocessor) egy nyílt forráskódú, számítógépes szkriptnyelv, legfőbb felhasználási területe a dinamikus weboldalak készítése. Emiatt a PHP-t jórészt szerver oldalon használják, bár létezik parancssori interfésze is, illetve önálló grafikus felületű alkalmazások is létrehozhatóak vele. A PHP a legtöbb webszerverre, operációs rendszerre és platformra ingyenesen telepíthető. Manapság több mint 20 millió weboldal és egymillió szerver futtat PHP-t, bár a nyelvet használó oldalak száma 2005 augusztusától kezdve folyamatosan csökken. A PHP emellett az Apache webszerver egyik legnépszerűbb beépülő modulja. A PHP működése leegyszerűsítve a következőképpen néz ki: az oldalak elkészítésénél a HTML-t gyakorlatilag csak mint formázást használják, ugyanis ezen lapok teljes funkcionalitása a PHP-re épül. Amikor egy PHP-ban megírt oldalt akarunk elérni, a kiszolgáló először feldolgozza a PHP utasításokat, és csak a kész (HTML) kimenetet küldi el a böngészőnek, így a programkód nem is látható kliens oldalról. Ehhez egy úgynevezett interpretert (értelmezőt) használ, amely általában egy külső modulja a webszervernek.

A PHP nyelv lényegében nagymértékű kiegészítése a HTML-nek, ugyanis rengeteg olyan feladat végezhető el vele, amelyet nélküle nem tudnánk elvégezni, vagy ha igen, akkor is csak korlátozottan. Ilyen például a bejelentkezés, adatbáziskezelés, fájlkezelés, kódolás, adategyeztetés, kapcsolatok létrehozása, e-mail küldése, adatfeldolgozás, dinamikus listakészítés stb. Minden olyan esetben, ahol nagyszámú ismétlődő feladatsort kell végrehajtani (például képek listázása és linkelése, listakészítés stb.), ott ez a programnyelv nagyszerű segítség.

2.2 JavaScript (kliens oldali programnyelv)

A JavaScript (rövidebben csak JS) programozási nyelv egy objektuorientált szkript nyelv, amelyet viszonylag sok oldalon használnak. Kezdetben a neve LiveScript volt és csak később kapta a JavaScript nevet. Ezek után szintaxisa is közelebb került a Java programozási nyelvhez. Fontos megjegyezni, hogy míg a PHP kód a szerver oldalon fut le addig a JavaScript a kliens oldalon. Ezáltal sokkal dinamikusabbá tudjuk tenni weboldalunkat, mivel nem kell minden változtatáshoz újratöltenünk az oldalt és elküldeni a szerver felé az információt. Ezzel kapcsolatban szeretnék hozni egy példát. Tegyük fel, hogy éppen egy jelentkezési űrlapot töltünk ki, vagy ha a szakdolgozati témámnál akarok maradni, akkor egy hallgatót szeretnénk bevinni a rendszerünkbe (lásd 1. ábra). Ekkor ugyebár minden mezőnek megvan az egyedi sajátos tulajdonsága. Például, hogy a Neptun-kód csak hat karakter hosszú lehet és csak az angol ABC betűit tartalmazhatja, valamint számjegyeket nullától kilencig. Tehát minden mezőnek megvan a formátuma és hogy hogyan kell kinéznie. Amikor már kitöltöttük az adatlapunkat és fel szeretnénk venni egy hallgatót a rendszerbe akkor általában egy gombra kell kattintanunk az adatlap alján, aminek a felirata „Elküld”, „Felvesz” vagy valami hasonló. Ha nincs JS akkor a gomb lenyomása után az adatokat elküldjük a szervernek, ő feldolgozza azokat és kapunk egy választ. Ez a válasz az adatok helyességétől függően sikeres vagy éppen sikertelen felvételt jelenthet. Gondoljunk csak bele, ha mondjuk nem jól adtuk meg a Neptun-kódot, akkor úgymond futottunk egy teljesen felesleges kört. Itt jön a képbe a JavaScript melynek segítségével még az elküldés előtt leellenőrizhetjük az adatlapunkat, hogy az megfelelően van-e kitöltve, és csak akkor engedi nekünk a rendszer elküldeni az adatokat, ha a feltételek teljesülnek. Sőt, ha hibásan van kitöltve az adatlapunk akkor még a hibákra is figyelmeztet minket.

Az 1. ábrán az általam készített rendszeren szeretnénk felvenni egy hallgatót, mint ahogyan azt a képen is látjuk a rendszer figyelmeztet minket arra, hogy néhány mezőt nem töltöttünk ki és a Neptun-kódnál érvénytelen karaktert adtunk meg.



1. ábra: Adatlap kitöltése JavaScript ellenőrzéssel

Viszont a JavaScripteknek van egy olyan hátulütőjük, hogy bár próbálják maximalizálni a platformfüggetlenséget mégis a különböző böngészők eltérő módon implementálják a kódot. Ezért százszázalékosan sosem bízhatunk meg a működésükben. Ha nincs bekapcsolva az adott böngészőben a JS támogatás, vagy nincs hozzá megfelelő támogatás, akkor az ellenőrzés nem fog végrehajtódni. Mivel a PHP-nál ilyenről nincs szó és mindenhol egységesen ugyanúgy működik, ezért fontos, hogy a kódunk duplán biztonságot nyújtson. Tehát a háttérben mikor megnyomjuk a gombot akkor egy szerver oldali ellenőrzésnek is mindig le kell futnia nehogy hibás adat kerüljön a rendszerbe. Persze ebből a felhasználó a JavaScript helyes működése esetén semmit nem fog tapasztalni és csak akkor fog lefutni az ellenőrzés, ha a JavaScripten már átmentek az adatok.

A JavaScripthez léteznek úgynevezett könyvtárak (libraries), ezek olyan függvény- és osztálykönyvtárak, melyek már előre meg vannak írva és funkciójuk szerint össze vannak gyűjtve. Ezek segítségével könnyebbé válik a JavaScript alkalmazások fejlesztése. Számos ilyen könyvtár létezik, én személy szerint az egyik legismertebbet választottam a jQuery-t és mellette néhol használtam az Ext JS-t is.

2.3 CSS (stílusleíró nyelv)

Az oldalak formázásához CSS-t (Cascading Style Sheets) használtam, melynek nevét nem nagyon szokták magyarra fordítani, de ha mindenképpen ragaszkodunk hozzá akkor stíluslapoknak nevezhetjük őket. Röviden annyit róluk, hogy egy stílusleíró nyelv mely a HTML vagy az XHTML típusú strukturált dokumentumok megjelenését írja le. A CSS kódunkat kétféleképpen adhatjuk hozzá a HTML kódunkhoz, az egyik, hogy direktbe az oldalba ágyazva írjuk a kódot a *<style type="text/css"> </style>* tagek közzé. Ezt nevezzük beágyazott stíluslapoknak. A másik lehetőség, hogy külsőleg hivatkozunk rájuk ily módon: *<link rel="stylesheet" href="kulso.css" type="text/css">* ekkor pedig külső stíluslapról beszélünk.

A nyelv felépítése nagyon egyszerű. A következőképpen néz ki:

kiválasztó { tulajdonság }
kiválasztó { tulajdonság }
kiválasztó { tulajdonság }

A kiválasztó rész azt határozza meg, hogy mely HTML elemekre vonatkozzon a definíció, míg a tulajdonság, hogy az adott elem milyen tulajdonságokkal bírjon, vagyis hogy nézzen ki. A HTML elemekre sokféleképpen hivatkozhatunk, ezeket én most nem fogom bemutatni részletesen, a két leglényegesebb az class és az id. A kettő között annyi különbség van, hogy míg az id-vel csak egy elemet tudunk megjelölni a HTML dokumentumon belül, addig a class-al többet is.

A CSS kódban a class-ra *.classnév,* illetve az id-ra *#idnév* módon hivatkozunk. Egy-egy példa az említett hivatkozásokra:

```
#header{
    position:relative;
    top:0px;
    margin:0px 10px 0px 10px;
    padding:0;
    background-color:#3399FF;
}
```

```
.form-label {
    float: left;
    width: 200px;
    text-align: left;
    margin-top: 4px;
}
```

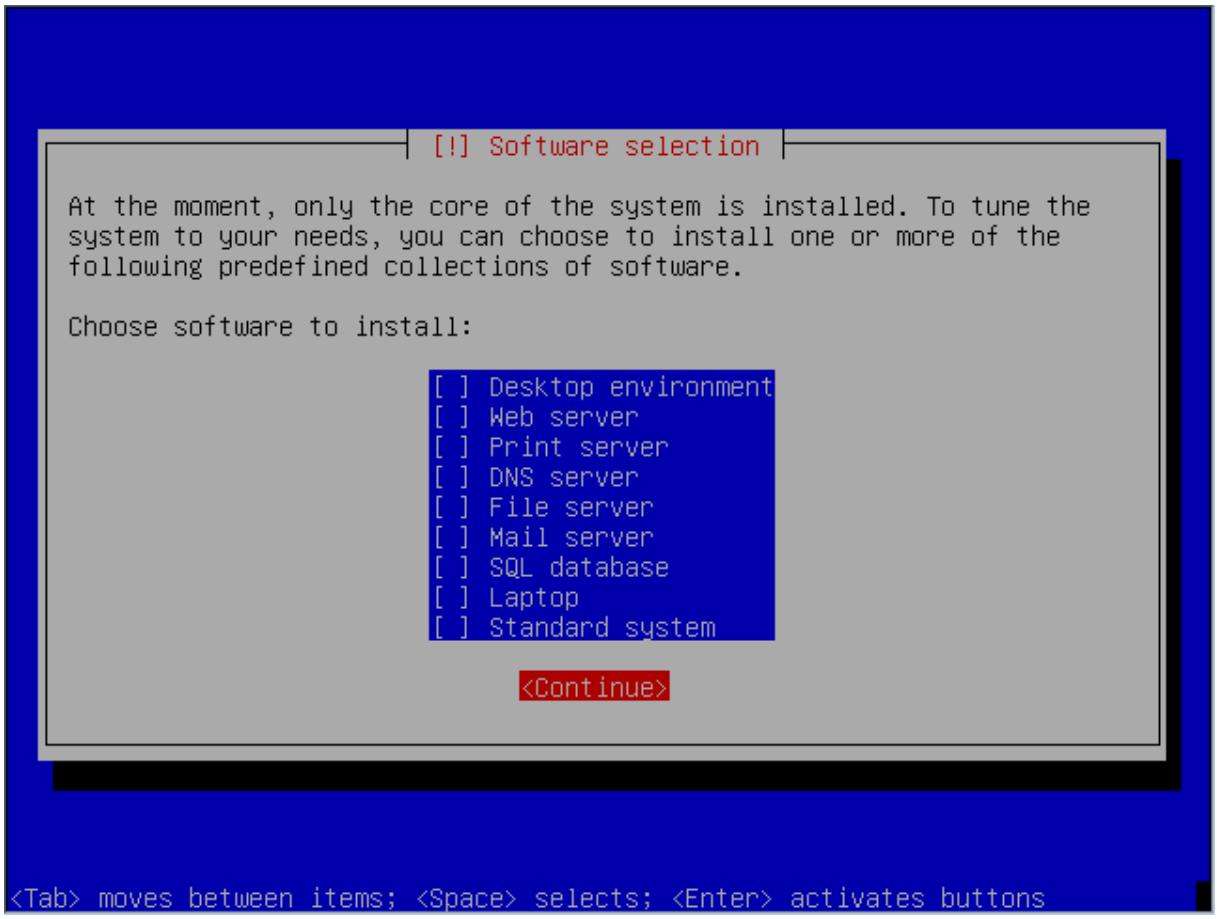
3. A rendszer beüzemelése

3.1 A szerver oldali rendszer kiválasztása

Mivel a webes alapú alkalmazás mellett döntöttem, melynek az okairól már korábban írtam, ezért szükség volt egy kiszolgáló rendszerre, amelyen a weboldalt elhelyezem. Tehát az első lépés az volt, hogy kiválasszam a megfelelő környezetet. Mindenképpen valamilyen Linux disztribúciót akartam választani, persze ez nem jelenti azt, hogy más platformokra például Windows-ra vagy akár OSX-re nem léteznek a megfelelő szoftverek. Mégis amikor az operációs rendszert kellett kiválasztanom, már tudtam, hogy nem ezekkel akarok dolgozni. Ennek az az oka, hogy a cégeknél a legtöbb rendszergazdai pozíció betöltéséhez szükséges valamilyen szintű Unix/Linux ismeret. Valószínűleg nem ennek a rendszernek az elkészítésével szereztem meg ezeket az ismereteket, de gyakorlásnak, illetve arra, hogy lássam, hogyan épül fel egy ilyen kiszolgálórendszer mindenképpen jó volt. Emellett a szempont mellett fontos volt számomra a stabilitás és a megbízhatóság is. Ezek után már csak azt kellett kitalálnom, hogy milyen disztribúciót válasszak. A döntés nem volt nehéz. Az általam legjobban ismert és a legjobbnak tartott rendszert választottam. Ez nem más, mint a Debian 5.0, vagy kódnevén Debian Lenny.

A két megnevezés ugyanazt a rendszert rejti magában, csak míg az egyiknél a verziószámot nevezzük meg, addig a másiknál a kódnevet. Ugyanis minden nem rolling release alapú rendszernek van egy kódneve, amely verzióról verzióra változik. Ez mind nagyon szép gondoljuk magunkban, de mit jelent az a rolling release. Lévén, hogy többfajta Linux disztribúció létezik, így a rendszerektől függően a fejlesztés menetében is eltérések lehetnek. Ez a gyakorlatban a következőképpen néz ki: a rolling release (ha nagyon magyarra akarnám fordítani akkor gördülő/guruló kiadás) disztribúcióknak nincs kötött verziószámuk, hanem a frissítések folyamatosan jönnek, míg a nem ilyen fajta disztribúcióknál a frissítések ciklikusan ismétlődnek, mondjuk hat havonta. Tehát a rolling release tipikusan gyakori kicsi frissítésekből áll, általában minden nap, vagy minden második nap van valamilyen kisebb újítás/javítás. A frissítéseket a csomagkezelő szállítja a felhasználó számára az interneten keresztül. Az ilyen rendszerek telepítése általában úgy szokott zajlani, hogy a disztribúció hivatalos oldalán mindig van egy „snapshot” ami egy naprakész képfájl az adott rendszerről, vagyis nagyon közel áll a legutóbbi frissítésekhez. Ezt a képfájlt kicsomagolva és egy DVD-re kiírva telepíthetjük a rendszert. Természetesen a telepítés után érdemes frissíteni, de mivel a snapshot nagyon közeli a legfrissebb állapothoz, ezért csak kevés újítást kell majd végrehajtani. Példa a rolling release disztribúcióra a Gentoo Linux, ami egy forrás alapú rendszer és a frissítések a fent említett módon zajlanak. Most már tudjuk miért van kódneve a Debian rendszereknek. Amellett, hogy szerintem ez az egyik legstabilabb Linux rendszer, nagyon elterjedt a szerver oldali rendszerek között, tehát a beüzemeléssel megszerzett tudás nagyon is piacképes és használható. Mivel a Debian rendszert főképp szerver oldalon használják ezért a frissítések itt nem hat hónaponként jönnek ki, hanem körülbelül két évente. Ennek az oka, hogy mivel a szervereknek magas rendelkezésre állást kell biztosítaniuk, ezért nem lehet őket folyton frissíteni, mert gondoljunk csak bele, ha gyakran jönnének ki az új verziók akkor nehézkes lenne mindig átmigrálni az új rendszerre, és emellett még a kiszolgálást is biztosítani. Másrészről a Debian fejlesztő csapata, csak akkor ad ki egy rendszert, ha az már bizonyos szinten stabilnak mondható, ezért működnek ezek a rendszerek ilyen jól ebben a környezetben. Mikor kiadnak egy új verziót akkor az előző támogatása, valamint a hibák javítása hatályos időn belül megszűnik. Amikor telepítjük a rendszert többek között kiválaszthatjuk, hogy milyen csomagok kerüljenek bele az alaprendszerünkbe (lásd a 2. képen).

Én csak egy csupasz operációs rendszert raktam fel, mivel saját magam akartam kiépíteni az infrastruktúrát. Természetesen a telepítés során még sok mindent lehet állítani, mivel én ezt a rendszert most csak fejlesztés céljából hoztam létre, nem a lehető legbiztonságosabb szervert építettem ki. Fontos megjegyezni, hogyha éles kiszolgálóról van szó, akkor legalább egy szoftveres raidet mindenképpen állítsunk be, illetve érdemes logikai kötetekben (LVM) gondolkozni, hogy a későbbi egy esetleges tárhely bővítés ne okozzon gondot.



2. ábra: A Debian operációs rendszer telepítése

Miután befejeztük a telepítést, az újraindítás után a Grub fogad minket. A Grub egy boot loader aminek a segítségével más telepített operációs rendszerek közül is választhatunk. Ez természetesen szervereknél ritka, mert ott csak egyvalami szokott telepítve lenni. Miután sikeresen felállt a rendszerünk és megkaptuk a bejelentkező felületet, máris elkezdhetjük az érdembeli munkát.

A telepítés során megadott név és jelszó párossal bejelentkezve az első dolog amit tennünk kell, hogy hozzáadjuk a felhasználónkat a /etc/sudoers fájlhoz. Ennek az az értelme, ha a későbbiekben, valami olyat szeretnénk csinálni, amihez rendszergazdai jogok kellenek, ne kelljen állandóan bejelentkezni rendszergazdaként, hanem szimpla felhasználóként is tudjunk parancsokat kiadni. Ez természetesen nem azt jelenti, hogy ezután már bárki rendelkezik rendszergazdai jogokkal, mert a jelszó megadása ekkor is kötelező. Egy ilyen parancs a következőképpen néz ki:
sudo végrehajtandó parancs. Tehát a felhasználónak ha rootként akar tevékenykedni akkor mindig a parancs elé kell írni a sudo szócskát. Az enter lenyomása után a rendszer kéri a root jelszavát, majd ha azt helyesen megadtuk, akkor végrehajtja az utasítást. Az első amit meg kell tennünk, természetesen csak a fentebb említett lépések után, a rendszer csomaglistájának frissítése, majd maga a rendszer frissítése. Ezek után most már tényleg egy teljes értékű szerver van a birtokunkba, amit elkezdhetünk a saját kedvünk szerint építgetni.

3.2 A szükséges csomagok bemutatása

3.2.1 Az Apache HTTP Szerver

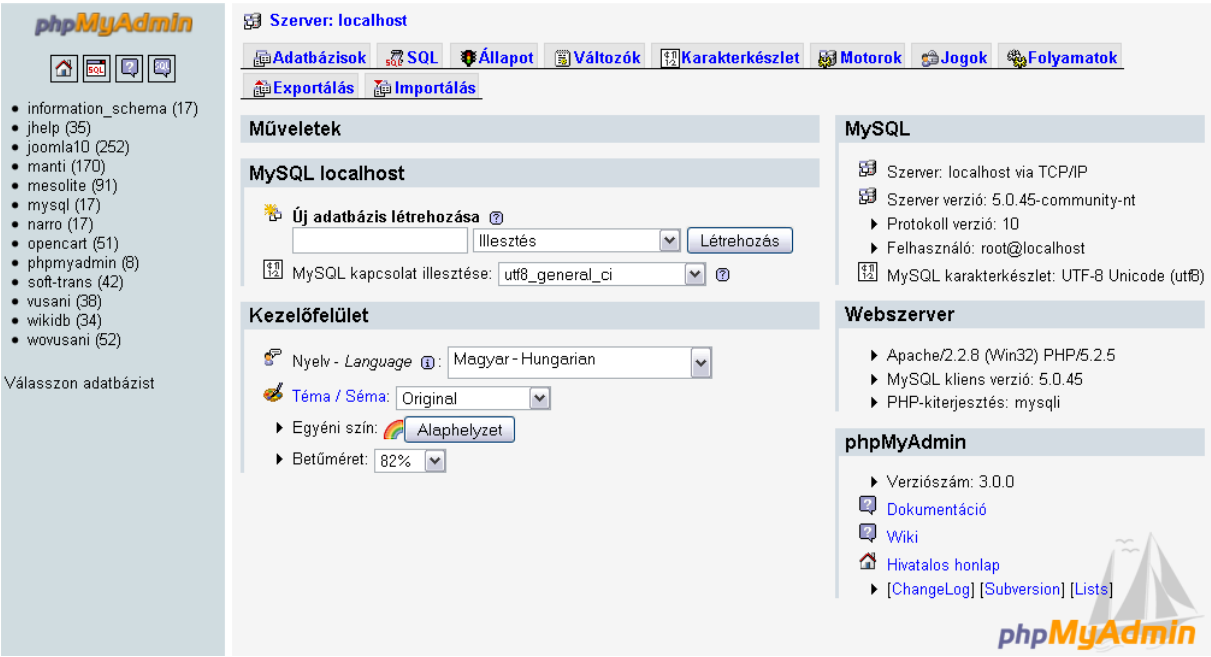
Az Apache HTTP Szerver, vagy ismertebb nevén csak Apache egy széles kőrben elterjedt webkiszolgáló. Az Apache volt az első életképes ellenfele a Netscape Communications Corporation vagy a ma ismert nevén Oracle iPlanet webszervernek. Azoknak a rendszereknek a többsége – melyek ezt a kiszolgálót futtatják – valamilyen Unix szerű rendszer. Természetesen ez nem azt jelenti, hogy az Apache csak ilyen rendszereken fut. A fejlesztők nagyon széles körben gondolkodtak amikor elkészítették ezt a szoftvert, ugyanis az Apache elérhető többek között Linux, FreeBSD, Solaris, Windows és OSX operációs rendszereken. 1996 óta ez a webkiszolgáló a legelterjedtebben használt HTTP szerver szoftver. 2010 februárjában az összes weboldal 54.46%-át szolgálta ki az Apache. Egyébként az Apache-ot C-ben írták és a legutolsó stabil verziója a 2.2.17-es. Számos beépülőmodult támogat a rendszer, a szerver oldali programozástól kezdve, a különböző autentikációs sémákig. Íme néhány programnyelv amit támogat az Apache: Perl, Python, Tcl, valamint az egyik legismertebb és az általam is használt PHP. Emellett az elterjedtebb autentikációs modulok a mod_acces, mod_auth és a mod_digest.

Nagyon fontos még megemlíteni az Apachne-nak azt a képességét, hogy lehetőséget ad egy rendszeren belül több weboldal futtatására is. Ezt a Virtual Hosting teszi lehetővé, ennek segítségével lehet nekünk több oldalunk is az adott szerveren. Ez által a www.pelda.hu és a www.valami.org is működni fog ugyanazon a kiszolgálón, és persze az ehhez tartozó aldomaineket is el tudjuk majd érni. Szerintem ennyi elég is lesz az Apache-ról elöljáróban, a dolgozatom során még úgyis fogok rá hivatkozni.

3.2.2 A MySQL adatbázis-kezelő rendszer

Adatbázis-kezelő rendszernek a MySQL-t választottam ez az egyik legelterjedtebb ilyen rendszer és nagyon széles körben alkalmazzák, valamint része a LAMP-nak (Linux-Apache-MySQL-PHP), ezért konfigurációja egyszerű és gyors. Lévén, hogy a MySQL egy SQL alapú rendszer, és én már az egyetem keretein belül tanultam az SQL nyelvről, ezért nem volt nehéz megismerkednem a szintaxisával, működésével. Másik nagy előnye ennek a rendszernek, hogy nagyon sok programnyelvből elérhető (C, C++, C#, Delphi, Java, PHP, Ruby, Python, Perl, stb.), ezért, ha a későbbiekben tovább szeretném fejleszteni a programomat, kiegészítve egy lokális verzióval, amely például C#-ban van megírva, ezt könnyen megtehetem. Így egy, vagy több adatlapot kitöltve és elmentve azt a megfelelő módon, nincs szükség folyamatos hálózati elérésre, hanem az adatokat később is feltölthetjük az adatbázisba. Természetesen nem csak én gondolom úgy, hogy a MySQL az egyik legjobb – ha nem a legjobb – ilyen jellegű rendszer. Sok olyan szerverezet használja ezt az adatbázis-kezelőt akiknek igen magas rendelkezésre állást kell biztosítaniuk a sok látogató miatt. A MySQL-t használók közül csak néhány nagyobb oldalt említenék meg. Wikipédia, Facebook, YouTube de akár mondhatnám a Google-t is, nekem ez elég referencia volt ahhoz, hogy én is ezt válasszam. Ez a rendszer C-ben és C++-ban íródott mint már említettem az SQL nyomán. Gondolom itt sem kell ecsetelnem a platformfüggetlenséget, ugyanis a MySQL is fut az Apache-nál említett rendszereken, de szerintem ez nem meglepő. Amit még érdemes megemlíteni a szoftverrel kapcsolatban, hogy hogyan érhető el. Ennek több módja van, én a legegyszerűbb parancssoros változatot használtam, ehhez nem kell semmit telepítenünk pluszba a MySQL mellé. Persze ennek meg van az a hátránya is, hogy semmilyen grafikus interfészt nem kapunk, mindent magunknak kell begépelni. A táblák létrehozása, módosítása, feltöltése is parancsok begépelésével zajlik.

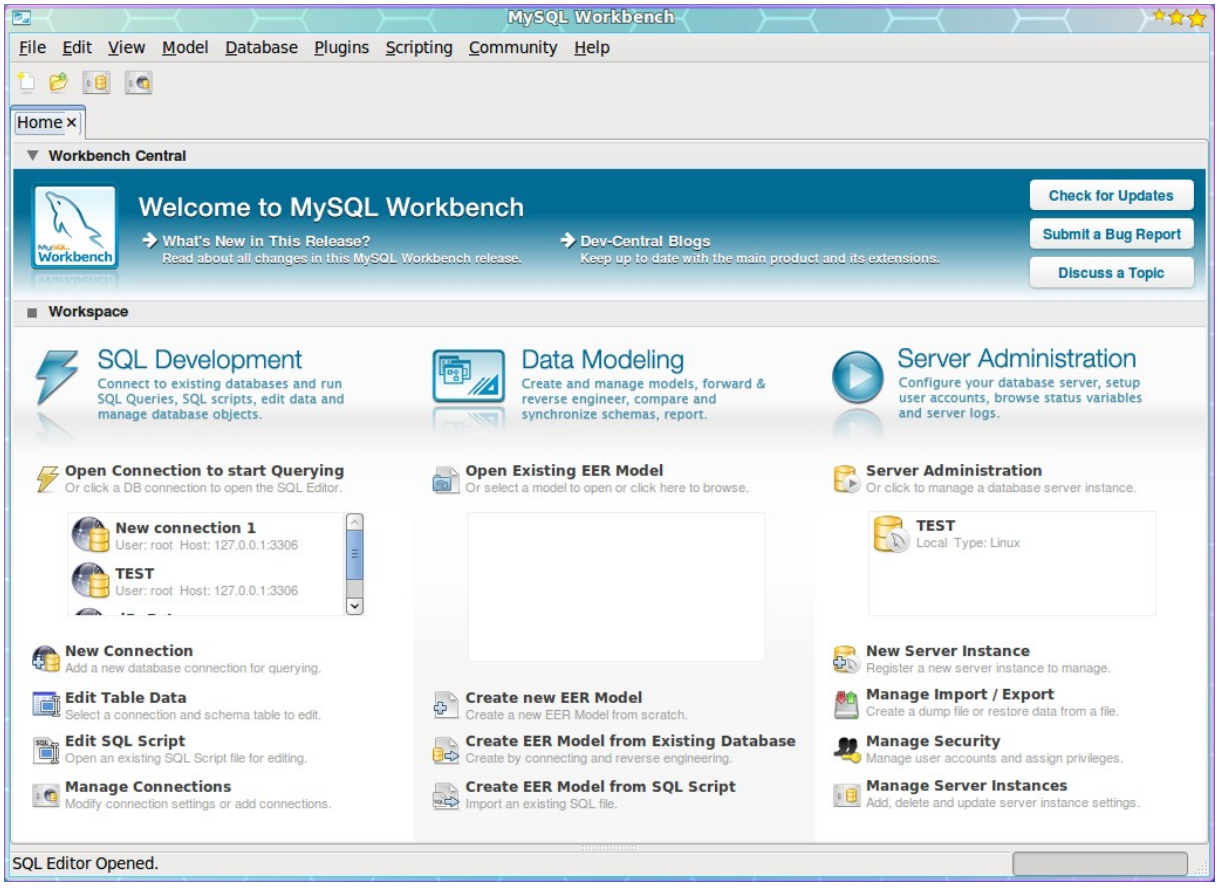
Ha ezt a módszert választjuk, nem árt otthon lenni kicsit mélyebben is a MySQL lelkületében, különben nehezen fogunk boldogulni. Ilyenkor a MySQL kezelése úgy néz ki, hogy valamilyen módon elérjük az operációs rendszerünk parancssorát, akár SSH-n keresztül, akár másképp. Kiadjuk a következő parancsot: *mysql -u felhasználó -p*. Ez a legegyszerűbb módja az adatbázis-kezelő rendszer elérésének, természetesen még léteznek más kapcsolók is mint például a -h amivel megadhatjuk a hostot (a szervert amin fut a MySQL), de általában csak a localhostról lehet belépni a rendszerbe, ennek biztonsági okai vannak. Miután megemlítettem a parancssoros módszert mindenképpen szeretnék még két alternatívát bemutatni. Az egyik és egyben legnépszerűbb is, a phpMyAdmin amely egy nyílt forrású eszköz. PHP-ban írták és arra való, hogy elérjük a MySQL felületét a weben keresztül, a telepítéséről majd lentebb fogok beszélni. Íme egy kép a felületről.



3. ábra: A phpMyAdmin felülete

A webes megoldásokon kívül léteznek még az operációs rendszerre telepíthető változatok is. Ilyen például a MySQL Workbench amely egy ingyenes szoftver, amit a MySQL fejlesztett ki. Ez a változat is hasonló funkciókra képes mint a fentebb említett phpMyAdmin, viszont ennek az elérési formának az az előnye, hogy nincs szükségünk telepített Apache webszerverre az adatbázisunk eléréséhez.

Bár ahol a MySQL előfordul, ott általában valamilyen webszerver is szokott lenni, így nem okoz problémát a phpMyAdmin telepítése, de lehetnek kivételes esetek. Természetesen itt is fenn áll az amit már említettem a bevezetőmben, hogy míg a phpMyAdmin-nál kliens oldalon nem kell semmilyen előkészületet tennünk, hogy elérjük az adatbázist, addig a MySQL Workbench-nél a programnak telepítve kell lennie az adott operációs rendszeren.



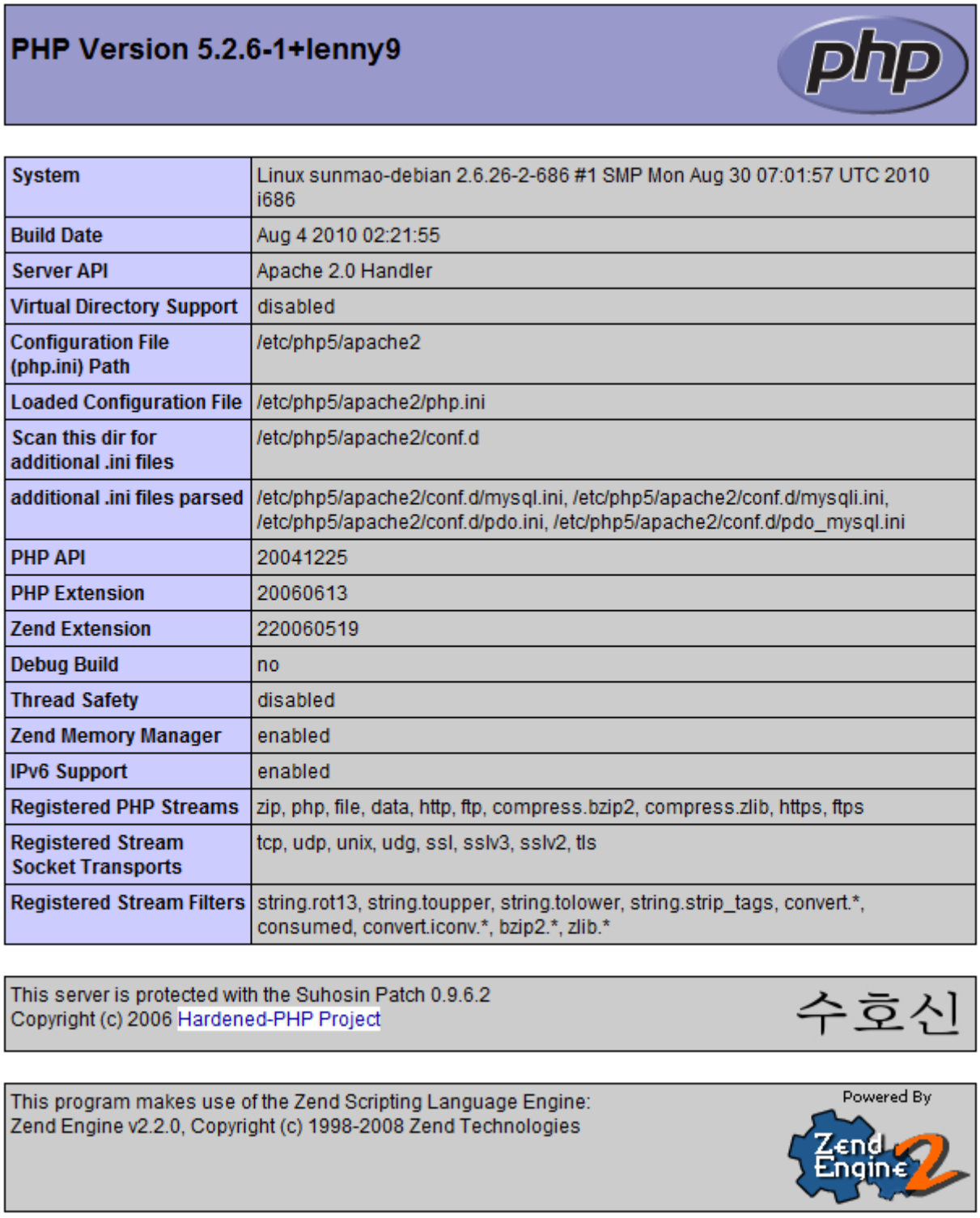
4. ábra: A MySQL Workbench felülete

Mint ahogyan az már fentebb említettem, a MySQL Workbench képes arra, mint a phpMyAdmin, de azért a fentebb látható képen érezzük azt, hogy ez azért mégis valamivel több, és komolyabb felület. Például a középen látható menüpontok (EER-el kapcsolatos műveletek) máris árulkodnak arról, hogy mivel is nyújthat többet ez a felület a többi társánál. Ami nagy előnye ennek a szoftvernek, hogy ingyenes és szinte bármilyen rendszeren elérhető.

3.3 A csomagok telepítése

Most, hogy bemutattam milyen csomagokra lesz szükségünk, szeretném bemutatni azokat a lépéseket amelyek segítségével beüzemelhetjük az általam írt programot. Itt már nem szeretnék azzal foglalkozni, hogy hogyan telepítsük az operációs rendszert, mivel erről nagyon sok leírás található az interneten és amúgy sem feladata ennek a dolgozatnak ezzel a résszel foglalkozni. Tehát, van egy frissen telepített Debian rendszerünk, amin a fentebb megbeszélt lépések alapján beállítottuk a sudoers fájlt. Most telepítenünk kell az Apache-ot, a hozzá tartozó modulokat és a MySQL adatbázis-kezelő rendszert. A csomagok telepítésében az aptitude nevű csomagkezelő rendszer lesz a segítségünkre. Megbeszéltük, hogy a telepítés után rögtön frissítjük a rendszerünket, de azért amikor egy újabb szoftvert szeretnénk telepíteni, akkor nem árt újra lekérni az aktuális frissítéseket. Tegyünk mi is így a *sudo aptitude update && aptitude upgrade* parancs kiadásával. A következő lépés a MySQL szerver és a hozzá tartozó kliens telepítése: s*udo aptitude install mysql-server mysql-client.* Amint látjuk installáláskor több csomagot is megadhatunk egyszerre, ehhez annyit kell tennünk, hogy a csomagneveket szóközzel elválasztva felsoroljuk egymás után (a sorrend nem számít). A telepítés során a rendszer meg fogja kérdezni, hogy milyen root jelszót szeretnénk beállítani a MySQL szerverünknek. Ennek semmi köze nincs az operációs rendszerben beállított root jelszóhoz. Ha a telepítéskor nem kérdezte volna meg a rendszer a jelszót, vagy utólagosan szeretnénk azt átállítani, bármikor megtehetjük azt a *sudo /usr/bin/mysqladmin -u root password 'rootjelszó'* parancs kiadásával. A szolgáltatásnak rögtön a telepítés után el kell indulnia. Azt, hogy fut-e a szerverünk, többféleképpen is leellenőrizhetjük. Az első és egyben legegyszerűbb módja az ellenőrzésnek a *ps axu | grep ^mysql* parancs kiadása, ha a parancsnak nincs visszatérési értéke, akkor valami baj van mert nem fut a szolgáltatás. Egy másik módszer a *sudo /etc/init.d/mysql status* parancs kiadása amely szerintem önmagáért beszél. Szemfülesebbek észrevehették, hogy az első módszernél nem írtam oda a parancs elé a sudo szót. Ez azért van, mert a futó processzeket kilistázhatjuk egyszerű felhasználóként is, nem kell hozzá root jogosultság. Ha nem fut a MySQL szerverünk, akkor valamit rosszul csináltunk, esetleg megpróbálhatjuk elindítani a *sudo /etc/init.d/mysql start* parancs segítségével. Ha nem kapunk hibaüzenetet, akkor nincs semmi probléma, de ha a telepítés után, nem indult el egyből a szolgáltatás akkor nagy valószínűséggel kapni fogunk valamilyen hibát.

Miután megbizonyosodtunk róla, hogy fut a szolgáltatásunk, az első és nagyon fontos dolog amit le kell szögeznünk, hogy soha ne lépjünk be root felhasználóként az adatbázisunkba. Ha ezt megszegjük annak nagyon komoly biztonsági következményei lehetnek. Ehelyett hozzunk létre egy felhasználót és ezzel érjük el az adatbázis rendszerünket. Ennek több módja is van, a legegyszerűbb módja a phpMyAdmin felületén bejelentkezve egy új felhasználó hozzáadása. Természetesen ilyen esetekben megszeghetjük a fentebb említett szabályt és beléphetünk root felhasználóként. Viszont semmilyen más esetben, vagy csak nagyon ritkán fordulhat az elő, hogy rootként lépünk be. Miután létrehoztuk a felhasználót a későbbiekben azzal jelentkezzünk be. Most, hogy ezzel készen vagyunk, telepítsük fel a webkiszolgálónkat amiről már annyit beszéltünk. Ehhez adjuk ki a *sudo aptitude install apache2 apache2-doc* parancsot. Most jön a P rész a LAMP-ból, ami nem más mint a PHP, *sudo aptitude install php5 php5-mysql libapache2-mod-php5* . Ezzel a paranccsal több dolgot is telepítünk, a php5 maga az interpreter ami majd le fogja fordítani a kódunkat. Emellett szükségünk van az Apache-nak azon moduljára ami a fordítót kezeli. Ez nem más mint a libapache2-mod-php. Itt szeretném megjegyezni, hogy ahhoz, hogy az általam írt program kifogástalanul működjön a PHP-nak az 5.2-es vagy annál újabb verziójára van szükség. A php5-mysql pedig lehetővé teszi, hogy direktbe csatlakozzunk a MySQL szerverünkhöz a PHP szkriptből, emellett lehetővé teszi még a PDO (PHP Data Object) használatát is, amiről majd a későbbiekben még beszélni fogok. Ezek után nincs más dolgunk, mint letesztelni az Apache-ot és a PHP-t. Ehhez hozzunk létre egy teszt.php fájlt a /var/www/apache2-default/ mappába a kedvenc szövegszerkesztőnkkel és írjuk bele a **<?php phpinfo(); ?>** sort, majd mentsük el. Most nyissuk meg a http://szerverunk-ip-je/apache2-default/teszt.php oldalt, ha jól dolgoztunk valami hasonlót kell kapjunk:



PHP Version 5.2.6-1+lenny9

System	Linux sunmao-debian 2.6.26-2-686 #1 SMP Mon Aug 30 07:01:57 UTC 2010 i686
Build Date	Aug 4 2010 02:21:55
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
additional .ini files parsed	/etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, bzip2.*, zlib.*

5. ábra: A rendszer tesztelése a **phpinfo()** függvény segítségével

Ezzel készen is vagyunk a szerver előkészítésével. Még egy dolgot talán érdemes lehet beállítani a MySQL konfigurációjában, ami nem más mint a STRICT_ALL_TABLES mód. Ez azért fontos, mert így nem tudunk érvénytelen adatokat felvenni az adatbázisba. Természetesen az adatok bevitele JS és PHP szinten is ellenőrizve van, de adódhatnak olyan ritka esetek amikor mégis sikerülhet a felhasználónak érvénytelen adatot megadnia. Például, ha dátumnál betűket adunk meg és ez a mód nincs bekapcsolva, akkor a MySQL a dátum mezőjébe a következőt fogja írni, 0000-00-00. Ez elég kellemetlen lehet mikor a táblázatunkat nézzük és nem értjük miért szerepel ott az a sok nulla. Léteznek még más módok is, de mi most azokkal nem foglalkozunk. A módokat többféleképpen be lehet kapcsolni, az egyik, hogy a MySQL indításakor a *–sql-mode=”STRICT_ALL_TABLES”* kapcsolót hozzáfűzzük a parancshoz. Ennél viszont létezik egy egyszerűbb mód, mégpedig az, ha a konfigurációs állományba kapcsoljuk be a módot. Ez Debian rendszereken a /etc/mysql/my.cnf alatt található. Ehhez a fájlhoz kell hozzáadnunk az *sql-mode = 'STRICT_ALL_TABLES'* sort. Majd újra kell indítanunk a MySQL szervert a *sudo /etc/init.d/mysql restart* paranccsal. Arról, hogy sikerült-e aktiválnunk a módot úgy győződhetünk meg a legkönnyebben, ha bejelentkezünk a parancssori felületen és kiadjuk a SELECT @@sql_mode; parancsot. Ez visszaadja nekünk a jelenleg futó módokat.

3.4 A MySQL felhasználó és a táblák létrehozása

Mivel a rendszerünk a MySQL adatbázisát fogja használni, ezért létre kell hoznunk a szükséges táblákat, valamint a felhasználót akin keresztül majd hozzáférünk a rendszerhez. Ezt mint már fentebb említettem megtehetjük valamilyen grafikus interfészen keresztül is, vagy parancssoros módban is. Én nem telepítettem semmilyen grafikus interfészt, hanem az utóbbi módszert választottam. Az első dolog amit meg kell tennünk, az az adatbázis létrehozása. Majd az új felhasználó felvétele a megfelelő jogokkal. Ehhez nyissunk meg egy terminált, vagy érjük el a szerverünket valamilyen felületen, például SSH-n keresztül és jelentkezzünk be rootként a következő módon: *mysql -u root -p* . Most, hogy bejelentkeztünk létre kell hozni az adatbázist amiben a tábláinkat szeretnénk tárolni. Itt szeretném megjegyezni, hogy minden MySQL utasítás után kell tennünk egy pontos vesszőt jelezve a rendszernek, hogy parancsunk itt véget ér. Tegyük meg ezt a *CREATE DATABASE themsy_db;* parancs kiadásával.

A themsy_db az adatbázisunk neve lesz. Azért themsy, mert ezt a nevet választottam a programnak, a thesis management system angol kifejezés nyomán. Ha létrehoztuk az adatbázisunkat, akkor hozzá kell adni az új felhasználót a rendszerhez. Mivel megbeszéltük, hogy rootként sosem, vagy csak nagyon ritka esetekben csatlakozunk a MySQL szerverhez. Ehhez a *GRANT ALL PRIVILEGES ON themsy_db.* TO 'themsy'@'localhost' IDENTIFIED BY 'jelszó' WITH GRANT OPTION;* utasítást használjuk. Nézzük meg kicsit részletesebben mit is jelent ez a sor. A GRANT kulcsszóval jogosultságokat adunk a felhasználónak, ez a mi esetünkben úgy néz ki, hogy a themsy nevű felhasználónak teljes körű jogosultságot adunk a themsy_db adatbázisra. Ami még lényeges nekünk az az IDENTIFIED BY utáni rész. Itt adhatjuk meg, hogy mi legyen a jelszó amivel majd az újonnan létrehozott felhasználó csatlakozni tud a rendszerhez. Ezek után már csak a GRANT OPTION részről nem tudjuk, hogy mit jelent. Nos ez csak annyit jelent, hogy az adott felhasználó más felhasználóktól elveheti azokat a jogosultságokat amivel ő maga is rendelkezik. Természetesen ennek a fordítottja is igaz, tehát adhat is jogokat más felhasználóknak, de csak olyan jogokat amivel ő is rendelkezik. Most, hogy létrehoztuk az új felhasználónkat nincs több tennivalónk rootként, ezért jelentkezzünk ki az *exit;* paranccsal. Majd jelentkezzünk vissza az újonnan létrehozott felhasználónkkal a már tanult módon, *mysql -u themsy -p* . Belépés után mindig ki kell választanunk az adatbázist amin dolgozni szeretnénk. Mivel nekünk, csak a themsy_db nevű adatbázisra vannak jogaink ezért válasszuk most ezt, ilyen módon *USE themsy_db;* Ezek után létre kell hoznunk az adatbázison belül a táblákat ahol majd tárolni fogjuk a különböző adatainkat.

A táblák létrehozása a következőképpen néz ki:

```
1  CREATE TABLE users
2      (
3       user_id INT           PRIMARY KEY   AUTO_INCREMENT,
4       logname VARCHAR(20)   UNIQUE,
5       logpass VARCHAR(32)
6      ) ENGINE=INNODB;
7
```

Első lépésként a felhasználók táblát hoztuk létre. Itt lesznek tárolva azok a személyek akik majd hozzáférhetnek a rendszerhez. Jogosan kérdezhetnénk, hogy miért nem a logname mező az elsődleges kulcsunk. Ennek az az oka, hogy a user_id mezőt össze fogjuk majd kötni egy másik táblával aminek manage_session lesz a neve, és ha egy felhasználót kitörlünk, akkor a manage_sessionből is törlődnie kell a megfelelő soroknak, ezért úgy gondoltam, hogy célszerű lenne ezt a mezőt megtenni elsődleges kulcsnak. Ami még érdekes lehet ebben a kódban a UNIQUE kulcsszó. Ezzel biztosítjuk azt, hogy a felhasználónév egyedi legyen. Természetesen ezt a PHP programban is leellenőrizzük, de nem árt ha duplán biztosítva vagyunk. Így ha már létező nevű felhasználót szeretnénk hozzáadni a rendszerhez és valamiért ezt a webes felület megengedné nekünk, még akkor is figyelmeztet minket a MySQL, hogy ez nem lehetséges. Már csak az AUTO_INCREMENT szúrhatja a szemünket, ez csak annyit jelent, hogy a user_id-t nem kell megadnunk explicit módon, hanem ez egy olyan érték lesz ami mindig automatikusan növekszik új felhasználó hozzáadása esetén. Azáltal ha kitörlünk egy felhasználót keletkezhetnek lyukak, de ez nem befolyásolja a rendszer működését.

Térjünk át a következő táblára.

```
1  CREATE TABLE students
2      (
3       students_id INT          UNIQUE       NOT NULL   AUTO_INCREMENT,
4       nk          VARCHAR(6)   PRIMARY KEY,
5       name        VARCHAR(40)               NOT NULL,
6       state       VARCHAR(50)               NOT NULL,
7       state_date  DATE,
8       depart      VARCHAR(50)               NOT NULL,
9       tform       VARCHAR(50)               NOT NULL,
10      title       VARCHAR(100)              NOT NULL,
11      leader1     VARCHAR(40)               NOT NULL,
12      leader2     VARCHAR(40),
13      email       VARCHAR(50)               NOT NULL,
14      phone       VARCHAR(20)               NOT NULL,
15      comment     VARCHAR(100)
16     ) ENGINE=INNODB;
```

Ezen a parancson nincs sok magyarázni való. Amit még esetleg ki lehet emelni, és az előzőnél nem tettem meg az az ENGINE=INNODB sor. Az INNODB egy tárolási eljárás, a MySQL alapértelmezetten a MyISAM-ot támogatja, ez nem lesz jó nekünk, mert nem támogatja a külső kulcsok használatát. Érdekesség lehet, hogy az első verzióknál még nem volt students_id oszlopom, aztán ahogyan haladtam a fejlesztéssel rájöttem, hogy mindenképpen szükségem lesz rá ha például valakinek a Neptun-kódját meg szeretném változtatni. A következő táblánk a már említett manage_session lesz.

```
1  CREATE TABLE manage_session
2      (
3       sess_id    VARCHAR(32),
4       user_id    INT         UNIQUE NOT NULL,
5       lastlogin  DATETIME,
6       query      TEXT        NOT NULL,
7       PRIMARY KEY (sess_id),
8       CONSTRAINT fk_userid FOREIGN KEY(user_id)
9       REFERENCES users(user_id) ON DELETE CASCADE
10     ) ENGINE=INNODB;
```

Ez a tábla némi magyarázatra szorul több szempontból is. Először is kezdeném azzal, hogy miért volt szükség erre a táblára. Mivel a rendszert úgy kellett megterveznem, hogy platformfüggetlen legyen ezért nem hagyhattam figyelmen kívül azt sem, hogy az egyes operációs rendszerek többféleképpen kezelik a sessionöket. Erről majd bővebben fogok írni a kompatibilitási problémák résznél. Ebben a táblában letároljuk az adott felhasználóhoz tartozó session azonosítót, a felhasználó azonosítóját, azt, hogy az illető mikor jelentkezett be utoljára, valamint, hogy milyen szűrési feltételeket adott meg mikor lekérdezte a hallgatókat a rendszerből. Ami eddig még nem szerepelt a táblák létrehozásánál az az utolsó előtti két sor. Itt csak annyit csinálunk, hogy létrehozunk egy megszorítást aminek a nevek fk_userid lesz. Természetesen bármilyen nevet adhatunk a megszorításunknak, de érdemes olyan nevet adnunk amivel utalunk az adott megszorításra. Először is meg kell adnunk a külső kulcsot, ez nálunk a user_id lesz a manage_session táblában. Majd meg kell adnunk, hogy mivel szeretnénk ezt „összekötni”. A REFERENCES kulcsszóval utalunk a másik tábla mezőjére. Meg kell adnunk a táblát és zárójelbe az oszlopot amivel össze szeretnénk kötni az aktuális tábla adott oszlopát.

A gyakorlatban ez úgy néz ki, hogyha van egy felhasználónk aki szerepel a users táblában, akkor ha ezt a felhasználót kitöröljük a rendszerből, a manage_session táblából is törlődni fog. Ezt fogja megvalósítani ez az utasítás. Ezen kívül még három táblát kell létrehoznunk. Mivel ezek nagyon hasonlóak egyben mutatom meg őket.

```
1  CREATE TABLE states
2      (
3       state  VARCHAR(50) PRIMARY KEY
4      ) ENGINE=INNODB;
5
6  CREATE TABLE department
7      (
8       depart  VARCHAR(50) PRIMARY KEY
9      ) ENGINE=INNODB;
10
11 CREATE TABLE traning_form
12     (
13      tform  VARCHAR(50) PRIMARY KEY
14     ) ENGINE=INNODB;
```

Az itt látható három tábla azt teszi lehetővé, hogy dinamikusan bővíthessünk a rendszerünket különböző elemekkel. Ez a három tábla is a fejlesztés egy bizonyos szakaszában jött létre, mivel eredetileg a kódba írtam bele a választható opciókat. Aztán rájöttem, hogy nem tudok megadni minden opciót, és úgy lesz a legjobb, ha ezt a feladatot a felhasználóra bízom. A states tábla jelöli a különböző státuszokat, mint például a hallgató leadta a tanulmányi osztályon a dolgozatát, vagy már felvette a Szakdolgozat 2 nevű tárgyat. Ezekhez az értékekhez megadhatunk dátumot is, de ez nem kötelező. Ha visszagörgetünk egy kicsit a dolgozatban láthatjuk, hogy mikor létrehoztuk a students táblát ott megadtunk egy olyan mezőt aminek a neve state_date volt és típusa DATE. Emellett még azt is észrevehetjük, hogy az adott mezőnél nem szerepel a NOT NULL kulcsszó, tehát megadása opcionális. A department tábla a szakokra utal, mint például MI, PTI, GI stb… A traning_form pedig a képzési tagozatra, úgy mint nappali, levelező stb… Új opciók felvételére vagy már meglévők törlésére a webes felületen a beállítások menüpont alatt lesz lehetőségünk. Fontos megjegyezni, hogy habár most jól el vannak különítve az egyes paraméterek egymástól, ez csak a szemléltetés miatt van.

A szkriptben amit majd mellékelni fogok a dolgozatomhoz ott az egyes szavak csak egy szóközzel vannak elválasztva egymástól. Természetesen ha a parancsokat bemásoljuk a MySQL felületére így is lefutnak, de biztosabb az ha a szokásos módon adjuk meg ezeket az utasításokat. Most, hogy létrehoztuk az össze szükséges táblát, már csak egy teendőnk maradt a MySQL részen belül. Ez pedig nem más, mint az első felhasználó létrehozása. Erre azért van szükség, mert máskülönben nem tudnánk bejelentkezni a felületünkre. Persze miután létrehoztuk az első felhasználót a többit már a webes felületen is létrehozhatjuk vagy akár törölhetjük őket a már fentebb említett beállítások fül alatt. Itt szeretném megjegyezni, hogy érdemes elsőnek egy admin nevű felhasználót megadni, majd a különböző felhasználóknak újabb fiókot nyitni. Fontos, hogy ezt az admin nevű felhasználót ne töröljük a webes felületen a rendszerből, mert ha véletlen kitörölnénk az összes felhasználót, ezzel még mindig vissza tudunk jelentkezni. Viszont, ha ezt is kitöröljük, akkor többé nem tudunk bejelentkezni a webes felületen. Ekkor kénytelenek leszünk kézzel a MySQL parancssorából újból létrehozni az első felhasználót. Nézzük hogyan adjuk hozzá az első felhasználót a rendszerhez. Ehhez az *INSERT INTO users (logname,logpass) VALUES ('admin',MD5('jelszó'));* parancsot kell kiadnunk. Az INSERT INTO users utasítással megadjuk, hogy szeretnénk beszúrni egy új felhasználót a users táblába, akinek a neve legyen admin és a jelszava pedig jelszó. Természetesen a jelszó szó helyett itt valamilyen biztonságos általunk is erősnek vélt jelszót kell beállítanunk. Az MD5 függvény pedig arra utal, hogy a táblában a jelszavunkat nem egyszerű szövegként szeretnénk letárolni hanem titkosítva. Ezzel a végére értünk a MySQL beállításának, most jöhet a rendszer felmásolása a szerverre, valamint a megfelelő jogok.

3.5 A PHP fájlok felmásolása a szerverre és a jogok beállítása

Ebben a részben valamilyen médián keresztül fel kell másolnunk a programot a szerverre, majd be kell állítanunk a jogosultságokat. Erre többféle megoldás is létezik. A szervereket általában el lehet érni FTP-n keresztül ezért ez is egy lehetőség. Ami talán ennél is jobb, ha SSH-n keresztül másoljuk fel az adatokat. Windows alól tipikusan a WinSCP nevű programot szokták használni erre a célra. Tehát bejelentkezünk a szerverre és felmásoljuk a könyvtárakat és fájlokat a megfelelő helyre. Az, hogy hova kell másolnunk a programunkat a szerver beállításaitól függ. Általános esetben, a /var/www mappába szoktak lenni a weboldalak, ha csak egy oldal fut az adott szerveren.

Ha több oldal is fut a kiszolgálón, akkor a VirtualHost beállításaitól függően kell a megfelelő helyre felmásolni a programunkat. Ha véget ért a másolás, már csak a jogosultságokat kell beállítani. Erre azért van szükség, hogy ne férhessen hozzá bárki a programunkhoz. Ugyanis a Linux rendszereken háromféle szintet különböztetünk meg egy fájlnál vagy egy könyvtárnál a jogosultságok szempontjából. Mégpedig a tulajdonosnak milyen jogai vannak, illetve akinek a csoportjába tartozik a fájl vagy mappa, és a többieknek mik a jogai. Mind a három szinten be lehet állítani három jogosultságot, ezek név szerint az olvasás, írás, futtatás. Ezek alapján én a következőket javaslom. Mivel az Apache szervert általában a www-data nevű felhasználó szokta futtatni, ezért a tulajdonosa a fájloknak és mappáknak legyen a root és tartozzanak a www-data csoporthoz. A jogok pedig úgy nézzenek ki, hogy a tulajdonosnak (tehát csak is a rootnak) lehessen mind a három opciót gyakorolnia, (olvas, ír, futtat) a csoportnak (www-data) pedig csak olvasnia és futtatni lehessen a fájlokat, a többiek pedig semmilyen jogot nem kapnak. Emellett megjegyzem, ha több weboldal fut a szerveren akkor ennél még jobb megoldás, ha létrehozunk az oldalunknak egy külön felhasználót, és ő rajta keresztül futtatjuk a webszerverünket és minden fájlt és mappát ami vele kapcsolatos, szigorúan az ő tulajdonába helyezünk. Ehhez viszont mélyebb ismeretek szükségesek, ezért ebbe most nem mennék bele. Ha valakit érdekel ez a módszer akkor keressen rá az interneten az **apache2-mpm-itk** kulcsszóra. Én most az első esetet szeretném részletesebben taglalni. Tehát felmásoljuk a szerverünkre a fájlokat és könyvtárakat a /var/www/ elérési út alá. Ekkor lesz nekünk egy /var/www/themsy könyvtárunk. A következő két paranccsal beállíthatjuk a fentebb tárgyalt jogosultságokat.

sudo chown -R root:www-data /var/www/themsy
sudo chmod -R 750 /var/www/themsy

A telepítés rész véget ért, most már elkezdhetjük használni a webes a felületet.

4. A rendszer bemutatása

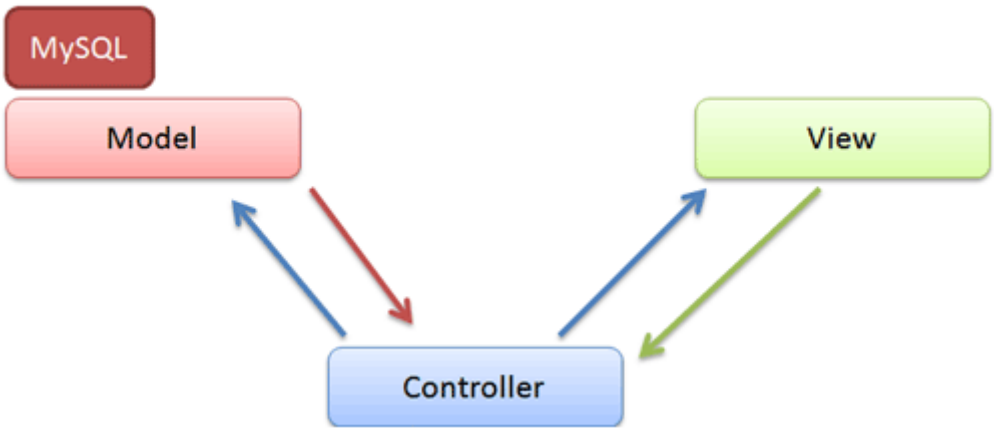
4.1 Programozási irányelvek

Ebben a fejezetben szeretnék kitérni egy kicsit az általam használt programozási mintákra, illetve a lehetséges implementációkra.

4.1.1 Az MNV

Mint ahogyan azt már korábban is említettem az MNV (Modell-Nézet-Vezérlő) egy szerkezeti minta. Mit is jelent ez valójában? Amikor egy ember vagy egy több személyből álló csapat egy programon dolgozik, akkor sok mindent figyelembe kell vennie mind a tervezésnél, mind a megvalósításnál. Az egyik ilyen fontos kívánalom, ha adatokkal dolgozunk és ezeket meg is kell jelenítenünk valamilyen formában, a modell és a nézet szétválasztása. Erre azért van szükség mert a felhasználói felületnek vagyis a nézetnek nem szabad befolyásolnia az adatkezelést. Emellett nagyon fontos kritérium, hogy az adatok átszervezhetőek legyenek anélkül, hogy a felhasználói felületen bármit is változtatnánk. Természetesen az adatokat valamilyen formában elérhetővé kell tenni a nézet számára. Itt jön a képbe a vezérlő, ugyanis ideális esetben a modell és a nézet direktbe nem kommunikál egymással, hanem az őket összekötő vezérlőn keresztül érik el egymást. Szeretném bemutatni egy konkrét példán keresztül, hogy ennek milyen haszna van a gyakorlatban, azon kívül, hogy az adatok átszervezhetőek lesznek a nézettől függetlenül. Maradjunk az én dolgozatomnál és vegyünk egy olyan esetet amikor egy hallgatót szeretnénk felvenni a rendszerbe. Ekkor ugye ki kell töltenünk megfelelő módon a felvételi lapot, majd el kell küldenünk az adatokat a szerver felé, ahol ezen adatok feldolgozásra kerülnek. Ilyenkor még végeznünk kell egy utólagos ellenőrzést abban az esetben, ha valamilyen oknál fogva a megadott adatok nem lennének érvényesek. Ennél a résznél jutottunk el oda, hogy miért hasznos az MNV séma használata. Ha használjuk ezt a módszert akkor mindössze annyi a feladatunk, hogy átpasszoljuk az adatokat a vezérlőnek, ő eldönti, hogy az adatok érvényesek-e. Ha igen akkor elküldi a modellnek ahol az adatok letárolásra kerülnek. Ha nem érvényes akkor a vezérlő küld egy választ a nézetnek, hogy valami baj van a megadott adatokkal. Ezt a felhasználó észleli és javítja a hibát.

Ezt persze meg lehetne valósítani az MNV használata nélkül is, de úgy egy sokkal bonyolultabb, nehezen átlátható kódot kapnánk. Arról nem is beszélve, hogy ez a módszer nagyban megkönnyíti a karbantarthatóságot és a kód hordozhatóságát. Most nézzük meg, hogyan is működik ez az egész egy keretrendszernél. Természetesen a különböző keretrendszerek másképpen értelmezhetik az MNV modellt és ebből következik, hogy annak implementációja is más lesz. Én most bemutatom ezen implementációk egy típusát. Megjegyezem, hogy jómagam a program elkészítésekor nem használtam semmilyen keretrendszert. Ennek az az oka, hogy szerintem ilyen kis oldalnál még nincs szükség egy teljes keretrendszer használatára. Persze azért én is próbáltam az MNV „stílusában" programozni.



6. ábra: Az MNV séma egy lehetséges értelmezése

Ezen a képen egy eléggé elterjedt keretrendszer MNV modelljét láthatjuk, bár a képnek van még egy része amit szándékosan nem mutattam meg, hogy a lényegre tudjunk koncentrálni. A keretrendszer nem más, mint az ismert CakePHP. Nézzük meg ezt a képet kicsit részletesebben. A view, magyarul nézet, az a rész amit a felhasználó lát, ezt jeleníti meg a böngésző. Az ábrán a nyilak jelzik a kommunikáció irányát. Észrevehetjük, hogy a modell és a nézet között nincsen nyíl, vagyis ők egymással direktbe nem kommunikálnak, csak a vezérlőn keresztül. A nézet nem is tud arról, mi folyik valójában a háttérben. Nézzük meg az előző példánkat, immáron az ábra segítségével. A nézetben elküldött adatot a vezérlő feldolgozza, vagyis elvégzi a szükséges ellenőrzéseket, átalakításokat.

Ezek után ha mindent jól adtunk meg, akkor az adatot továbbküldi a modellnek aki már közvetlenül tud kommunikálni az adatbázissal amely esetünkben egy MySQL rendszer. Az módosítások után a modell visszaküldi az eredményt (ha van ilyen) a vezérlőnek, aki pedig tovább küldi azt a nézetnek, ahol az új adatok megjelennek.

4.1.2 Az Egyke (Singleton) tervezési minta

Az Egyke vagy angolul Singleton tervezési mintának már a neve is utal arra, hogy az adott osztályból csak egyetlen példányt hozhatunk létre. Természetesen, ehhez arra van szükség, hogy a programunk megvalósítsa az objektumorientáltságot. A PHP nem szabja meg, hogy milyen módszerrel programozunk. Tehát a Javaval ellentétben itt nem vagyunk rákényszerítve az objektumorientáltságra. Ennek ellenére én mégis azt javaslom, hogy ha PHP-ban szeretne valaki programozni, akkor mindenképpen objektumorientáltan tegye azt. Ennek több oka is van. Az első és egyben legfontosabb szempont a kód magas fokú újrahasznosíthatósága. A tervezési minták viszont már túlmutatnak a kód újrahasznosításon. Ezek olyan előre kidolgozott megoldások, amelyeket bármikor bevethetünk és nagyon nagy segítséget nyújtanak a programozónak. Ezen fogalmak használata egységet teremt a programozók között, ezzel megkönnyítve a programozást. Visszatérve az Egyke tervezési mintára, mint már fentebb említettem, az adott osztályból a program futásakor egyetlen példány jön létre. Miért van erre szükség? A válasz nagyon egyszerű. Vannak olyan esetek amikor az adott objektumból felesleges, sőt egyenes zavaró lenne több példányt létrehozni. Erre egy nagyon jó példa lesz a MySQL kapcsolat felépítése. Vegyük például a következő kódot.

```
1  class Database
2  {
3      public function __construct() { ... }
4      public function connect() { ... }
5      public function query() { ... }
6      ...
7  }
```

Ez az osztály fogja megteremteni a kapcsolatot a modell és az adatbázis között. Akármikor amikor csatlakozni szeretnénk az adatbázishoz, létre kell hoznunk egy példányt a fent látható osztályból. Mivel általában egy program működése során sokszor végez műveleteket az adatbázison (SELECT, INSERT, UPDATE, stb) ezért minden alkalommal egy új Database objektumot hozunk létre. Ebből az következik, hogy mindig újabb és újabb kapcsolat fog felépülni. Ez viszont felesleges és memóriaigényes művelet. Minek minden alkalommal, létrehozni egy új kapcsolatot, mikor valamilyen műveletet végre szeretnénk hajtani az adatbázisunkon? Természetesen létezhetnek olyan esetek, amikor több példányra is szükségünk van az adott osztályból, de ez most nem ilyen lesz. Erre az igen elterjedt problémára az Egyke programozási minta lesz a megoldás. Most bemutatok egy olyan osztályt, ahol ezek alapján lett megírva a kapcsolódás az adatbázishoz. Majd elmondom, hogy mit miért csináltunk.

```php
<?php
defined ( '_EXEC' ) or die ( '403.14 - Belepes megtagadva.' );

class DB_class
{
    private static $instance = NULL;
    public static function getInstance ()
    {
        if (! self::$instance) {
            self::$instance = new PDO(
            "mysql:host=localhost;dbname=themsy_db",
            'themsy', 've75uDaF');
            self::$instance->setAttribute(
            PDO::ATTR_ERRMODE,
            PDO::ERRMODE_EXCEPTION);
        }
        return self::$instance;
    }
}
?>
```

Az első lépés ahhoz, hogy az Egyke mintát alkalmazni tudjuk az, hogy létrehozunk egy **instance** nevű property-t. Itt fogjuk tárolni az egyetlen példányát a DB_class osztálynak.

Megjegyzem, hogy ennek az **instance** property-nek statikusnak kell lennie, amit a static kulcsszóval jelölünk. Ami az én kódomban nem látható, de mindenképpen szeretném megemlíteni az a __constructor() priváttá tevése. Ezzel megakadályozzuk, hogy kívülről példányosítani lehessen az osztályunkat. Tehát a myDatabase = new DB_class() kódrészlet többé nem fog működni. A következő lépés, hogy írnunk kell egy metódust, amely létrehozza az egyetlen példányunkat és vissza is tér vele, hogy használni tudjuk. Ez lesz a mi példánkban a **getInstance()** metódus. Ez a furcsa kinézetű függvény lesz a felelős az egyetlen példányunk kezeléséért. Első lépésben a metódus, vizsgálja az **instance** property-t. Ha nincs még létrehozva példány, akkor elvégezi a példányosítást a konstruktor segítségével. Emlékezzünk arra, hogy a konstruktorunkat priváttá tettük, tehát csak az osztályon belül hívhatjuk meg. Mivel most az osztályon belül dolgozunk ennek semmi akadálya. A példányosítás után beállítja a PDO hibajelentőjét. Ennek semmi köze az Egyke programozási mintához, csak itt most erre is szükség volt. Ezek után visszatér a példányunkkal. Most már elérhetjük a DB_class példányunkat akárhonnan a programon belül. A következő módon hajtunk végre MySQL utasítást a Singleton minta segítségével.

```
1  public static function getDepart($type) {
2          $get = "SELECT depart FROM department";
3          try {
4              $result = DB_class::getInstance ()->query ( $get );
5          } catch ( PDOException $e ) {
6              echo $e->getMessage ();
7          }
8          foreach ( $result as $row ) {
9              $depart = $row ['depart'];
10             echo "<option value='$depart'>$depart</option>";
11             }
12
13     }
```

Ebből nekünk a **$result = DB_class::getInstance ()->query ($get);** sor mi igazán érdekes. Észrevehetjük, hogy nem hozunk létre külön példányt, hanem direktbe meghívjuk a **getInstance()** függvényt az osztályon keresztül.

4.1.3 A PDO használata

Akik programoztak már PHP-ban és a programjuk kapcsolódott is valamilyen adatbázishoz, azok biztosan elgondolkodtak már azon, hogy jó lenne valamilyen egységes felület amely segítségével, programjukat akár egy sor átírásával is átültethetnék MySQL adatbázis rendszerről például PortageSQL-re., vagy esetleg csak szerettek volna „tisztább", érthetőbb kódot. Szerencsére a PHP 5.2-es verziójától kezdve ez a probléma is megoldódott. Ugyanis innentől kezdve kapunk egy „bővítményt" melynek neve PDO (PHP Data Object). A használatával nagyban megkönnyíthetjük a saját munkánkat. Sokkal átláthatóbb és hordozhatóbb kódot kapunk, emellett számos hasznos beépített függvénye van, mint például a **fetchAll()** amely visszatér egy tömbbel melynek elemei a lekérdezés sorai lesznek. A rendszer maga, három osztályból áll: PDO, PDOStatements, PDOException. Az előbb említett függvény a PDOStatements osztályba tartozik. Az első és talán legfontosabb osztály a PDO az adatbázishoz való kapcsolódást menedzseli, valamint a műveletek végrehajtását az adatbázison. Legfontosabb függvényei a **query(), prepare(), exec(), commit()**. A PDOStatements osztály a már végrehajtott lekérdezések eredményének a kezelésére szolgál. A jelentősebb függvényei a **fetch(), fetchAll(), rowCount()**. A PDOException osztály szerintem magáért beszél, a hibakezelést valósítja meg. Mint ahogyan azt a fentebbi kódrészletben is láthatjuk, lekérhetjük a hibaüzenetet, de akár a hiba kódját és még sok minden mást is. Véleményem szerint mindenképpen megéri a PDO osztályt használni, az Egyke mintával együttműködve nagyon hatékony és elegáns megoldást kapunk. A kódunk sokkal átláthatóbb és kezelhetőbb lesz, és természetesen, a már annyit emlegetett hordozhatóság sem utolsó szempont.

4.2 A felület bemutatása

Ebben a részben a program felületét szeretném bemutatni. Tulajdonképpen öt részre van tagolva a rendszer. Ezek a részek név szerint a „felvesz", „listáz", „szerkeszt", „töröl", „beállítások". Amikor elkezdtem tervezni a felületet akkor még nem gondoltam, hogy ilyen részletesen szét fogom szedni a különböző opciókat. Csak egy „Felvesz" és egy „Szűr/Módosít" menüpontot akartam létrehozni. Ám ahogy haladtam a program írásával egyre több funkciót kellett beletennem abba, és már nem volt elég a kétfajta mód.

Mikor befejeztem a programot addigra mindegyik opció külön menüpontba került és a „felvesz” és „beállítások” menüpontot kivéve mindenhol lehet akár kézzel is szerkeszteni a MySQL utasításokat. Ez nagyon nagy előny, ha kicsit összetettebb parancsot szeretnénk kiadni. Gondoltam azokra is akik annyira nincsenek otthon az informatika világában. Ezért például egy SELECTet nem csak kézzel lehet begépelni, hanem „össze is lehet kattintgatni”. Természetesen az utóbbi módszerrel nem tudunk olyan profi utasításokat előállítani melyek képesek akár feltételek közötti kapcsolatok megadására is, de az egyszerűségnek ára van. Véleményem szerint, azért nem lehetett megoldani azt, hogy adattagok közötti kapcsolatot is lehessen „kattintgatással” állítani, mert annyiféle variáció létezhet, hogy azt én mint programozó nem tudtam volna lefedni. Ennél sokkal egyszerűbbnek bizonyult az általam használt megoldás. Aki összetettebb utasításokat akar az úgyis valószínűleg érteni fog az adatbázis kezeléséhez, a többieknek meg még mindig ott van a másik módszer.

4.2.1 A „felvesz” menüpont

Ezt a menüpontot már láthattuk az első ábrán. Itt tulajdonképpen hallgatókat lehet felvenni a rendszerbe. A csillaggal jelölt mezők kitöltése kötelező, ha valamilyen kötelező adatot nem adunk meg, vagy nem jól adunk meg, akkor a rendszer figyelmeztet minket és nem engedi felvenni az adott hallgatót az adatbázisba. A dolgozatom első felében is szó volt már arról, hogy nem csak JavaScript segítségével ellenőrzi a program az adatokat, hanem egy PHP szűrön keresztül is átmennek mielőtt véglegesen elmentődnének az adatbázisba. Viszont a hibaüzenetek megjelenítésért a JavaScript a felelős. A jQuery könyvtár segítségével, könnyen írhatunk űrlap (form) ellenőrző programot. Ami mégis nehézséget okozott nekem, az a dátum érvényességének ellenőrzése volt. Erre sajnos a beépített függvény nem volt alkalmas így saját függvényt kellett írnom. Ezek után a dátum megadási formát kedvem szerint alakíthattam. Úgy döntöttem, hogy a beviteli forma év, hónap, nap legyen kötőjellel elválasztva. Ami még érdekesség lehet ezen a menüponton belül az a Neptun-kód ellenőrzése. Ugyanis erre is saját függvényt kellett írnom, mivel nyilvánvalóan egy nem magyar fejlesztésű JS könyvtárban erre nincs beépített lehetőség. Egyéb érdekesség, hogy a státusz kiválasztása után az űrlaphoz dinamikusan hozzáadódik egy már fentebb említett dátum mező is.

Itt opcionálisan megadható dátum az előbbi formában, de lehetőség van arra is, hogy egy kis ikonra kattintva egy felugró naptárból válasszuk ki a megfelelő időpontot. Ez természetesen az összes többi menüpontnál is így működik.

4.2.2 A „listáz” menüpont

Ezen fül kiválasztása után szintén egy űrlapot fogunk látni, annyi különbséggel, hogy két részre fog oszlani az oldal. Az egyik lesz a már említett űrlap a másikban pedig lesz egy úgynevezett szerkesztő felület, ahol saját utasításokat is megadhatunk. Ha nem szeretnénk saját utasításokat megadni, akkor egyszerűen annyi a teendőnk, hogy az űrlap kitöltésével megadjuk azokat a tulajdonságokat amelyek alapján szűrni szeretnénk, majd betöltjük az utasítást a szerkesztő felületbe és lefuttatjuk a program által generált utasítást. Habár ezek után még mindig van lehetőségünk a szűrés feltételein változtatni, akár kézileg, akár az űrlap értékeinek változtatásával. Ha úgy gondoljuk, hogy megfelelően összeállítottuk a lekérdezésünket, akkor nincs más dolgunk mint lefuttatni a parancsot. Ehhez rá kell kattintanunk a futtatás gombra és megvárni míg megjelennek a találatok. A rendszer figyelmeztet minket, hogy ha nem adtunk meg semmilyen értéket sem kézileg, sem az űrlapon keresztül, akkor a szűrés úgy hajtódik végre, hogy az összes rendszerben lévő hallgatót fogja találatul adni. Miután lefuttattuk a szelekciónkat, a rendszer egy táblázattal fog visszatérni, melyben a szűrés feltételeinek megfelelően fognak a hallgatók megjelenni. A táblázaton belül beállíthatjuk, hogy hány hallgató jelenjen meg egyszerre, illetve ha találatok száma nagyobb mint az előbb említett érték, akkor lapozhatunk a találatok között. Ha szeretnénk új szűrési feltételeket megadni, akkor nincs más dolgunk, mint az oldal tetején található vissza nyílra kattintani, amely vissza vezérel minket a szűrő felületre. Attól függően, hogy milyen adatokat szeretnénk látni a táblázatban, a táblázat fölött az „Oszlopok megjelenítése” ablakban az oszlop neve melletti négyzetre kattintva ki/be kapcsolhatjuk az adott oszlop megjelenítését. Ha nincs szükségünk ezen beállításokra, akkor egyszerűen az ablak jobb felső sarkában a nyílra kattintva felgördíthetjük az ablakot, a legördítés hasonlóan zajlik. Amit még érdemes megjegyezni az a táblázat fölött található két gomb, mely nem más, mint a törlés és a nyomtatás. Ha szeretnénk egy vagy esetleg több hallgatót is kitörölni a rendszerből, akkor a táblázatban a sor végi jelölőnégyzetet kell kipipálnunk azoknál a hallgatóknál akiket el akarunk távolítani a rendszerből.

Egy másik lehetőség a CTRL billentyűt folyamatosan lenyomva tartva kattintunk az adott soron belül bárhol, ezzel kiválasztva a törölni kívánt hallgatót vagy hallgatókat. Ha megvagyunk a kijelölésekkel akkor a töröl gombra kattintva a rendszer megkérdez minket, hogy biztosak vagyunk-e a dolgunkban, és ha igennel válaszolunk akkor eltávolítja a kijelölt hallgatókat a rendszerből. A nyomtatás gomb szerintem önmagáért beszél. Annyit szeretnék megjegyezni vele kapcsolatosan, hogy csak a megjelenített oszlopokat fogja a rendszer kinyomtatni, de mielőtt az adatok nyomtatásra kerülnének, kapunk egy nyomtatási képet egy külön ablakban ahol még vissza tudjuk vonni a folyamatot. A táblázatban ha egy hallgatónak nem férnek ki az adatai akkor megnyithatjuk az adott tanuló adatlapját külön is. Ehhez duplán kell kattintanunk a hallgató sorára, ekkor megjelenik egy új ablak, ahol kényelmesen megnézhetjük az adott személlyel kapcsolatos adatokat. Mindemellett ebben a felbukkanó ablakban lehetőségünk van a hallgató adatainak módosítására is. Miután átírtuk az adatokat a mentés gombra kattintva a rendszer rögtön frissül. A táblázatban már az új, módosított adatok fognak megjelenni. Jogosan tehetjük fel a kérdést, akkor miért van szükség a „szerkeszt” és a „töröl” menüpontokra. A válasz az, hogy azért, mert itt nem tudunk tulajdonságok alapján adatokat módosítani, illetve hallgatókat törölni. Nincs lehetőség arra, hogy például egyszerre több hallgatónak is megváltoztassuk az egyes attribútumait. Vegyünk egy egyszerű, de életszerű példát. Azt a feladatot kapjuk, hogy módosítsuk azon mérnök informatikus hallgatók státuszát akik már felvették a Szakdolgozat 1 nevű tárgyat. Ezt ugyan megtehetnénk úgy is, hogy az összes olyan hallgatóra rákattintunk akik megfelelnek a kritériumoknak és átírjuk az adataikat, de ez egy nagyobb adatbázisnál igen hosszadalmas munka lenne. A törléssel kapcsolatban, megint csak egy szemléletes példával fogok élni. Töröljük azokat a hallgatókat akik ebben a félévben adták le a szakdolgozatukat. A példa magáért beszél, mert valljuk be, hogy ezt egyenként kibogarászni a táblázatból nagyon nehézkes lenne. Ehelyett az űrlap és az utasításszerkesztő felület segítségével kényelmesen megoldhatjuk az adott problémát.

4.2.3 A „szerkeszt” menüpont

Itt fogjuk tudni a már rendszerben szereplő hallgatóink adatait módosítani. A felület két űrlapból áll. Az elsőben megadhatjuk milyen új értékeket szeretnénk beállítani a feltételnek megfelelő tanulóknál. Ebből logikusan következik, hogy a második űrlapnak mi lesz a feladata. Megadhatjuk azon feltételeket amelyek alapján adatmódosítást szeretnénk végezni.

Itt már nincsenek olyan szigorú megszorítások mint a hallgató felvételénél, nem kötelező kitölteni szinte a teljes űrlapot, de azt megköveteli a rendszer, hogy legalább egy-egy adatot adjunk meg. Ha ezt nem teljesítjük, akkor figyelmeztet minket betöltéskor, hogy sajnos nem a kritériumoknak megfelelően töltöttük ki az űrlapot ezért az utasítást nem menti át a szerkesztő dobozba. Természetesen azért arra figyelnünk kell, hogy az új adatok amiket meg adunk érvényesek legyenek. Gondolok itt a Neptun-kódra és az E-mail címre. Ha ezeket nem megfelelően adjuk meg a rendszer megint csak nem fogja engedni a betöltést. Erről a menüpontról nem is szeretnék többet beszélni, mivel nem hiszem, hogy további magyarázatra szolgálna a felülete.

4.2.4 A „törlés” menüpont

Erről a menüpontról sem szeretnék hosszasan beszélni. Ezt a részt volt a legegyszerűbb elkészíteni, ugyanis itt csak annyira van szükség, hogy megadjuk milyen feltételek alapján szeretnénk törölni. A felület a már jól megszokott űrlapból és szerkesztődobozból áll. A betöltött adatok után a futtatásra kattintva a rendszer kér egy megerősítést, majd ha erre igennel válaszolunk, akkor a program visszatér a törölt hallgatók számával, ha nem volt olyan tanuló akire igaz lett volna a feltétel, akkor ez a szám nyilvánvalóan nulla lesz. A rendszer ezt is jelzi nekünk.

4.2.5 A „beállítások” menüpont

Ez egy extra menüpont amelyre a tervezés során még egyáltalán nem gondoltam. Most viszont, hogy már teljes egészében látom a felületet egyértelműen hasznos volt ezt a részt is elkészíteni. Ha elsőnek lépünk be a webes felületre a teendőinket mindenképpen itt kell elkezdenünk. Az első lépés amit meg kell tennünk az a felhasználók létrehozása akik később majd hozzáférhetnek a programhoz. Itt lehetőségünk van többek között erre is, illetve új opciók megadására. Az „Adminisztrátor létrehozás/törlése” ablakban adjuk meg a leendő adminisztrátorainkat és mentsük el őket. A rendszer figyel arra, hogy már felvett felhasználókat ne tudjunk még egyszer hozzáadni. Ha ezzel megvagyunk adjuk hozzá a programhoz a választható opciókat, amelyeket majd később akár el is távolíthatunk, vagy ha éppen arra van szükség bővíthetünk.

Mindenképpen meg kell adni minden tulajdonságnál legalább egy-egy értéket, mert különben nem fogunk tudni hozzáadni a rendszerhez hallgatót. Ugyanis ezek az opciók kötelezően kitöltendő adatok lesznek és tudjuk, hogyha nem adjuk meg a kötelezően kitöltendő adatokat, akkor a rendszer nem enged új hallgatót felvinni. A program általános használata során nem lesz többet szükségünk erre a menüpontra. Természetesen, ha a későbbiekben szükségünk lesz új felhasználó hozzáadására vagy már meglévő törlésére, illetve opciók hozzáadására vagy törlésére, ide visszatérve újfent megtehetjük ezeket. A menüpontok bemutatása itt véget ért. Azt hiszem részletesen kitárgyaltam az összes lehetőséget.

Ezzel el is érkeztem a dolgozatom utolsó fejezetéhez, ahol szeretnék kitérni arra, hogy a program elkészítése során milyen akadályokba ütköztem és hogyan sikerült ezekkel megbirkóznom.

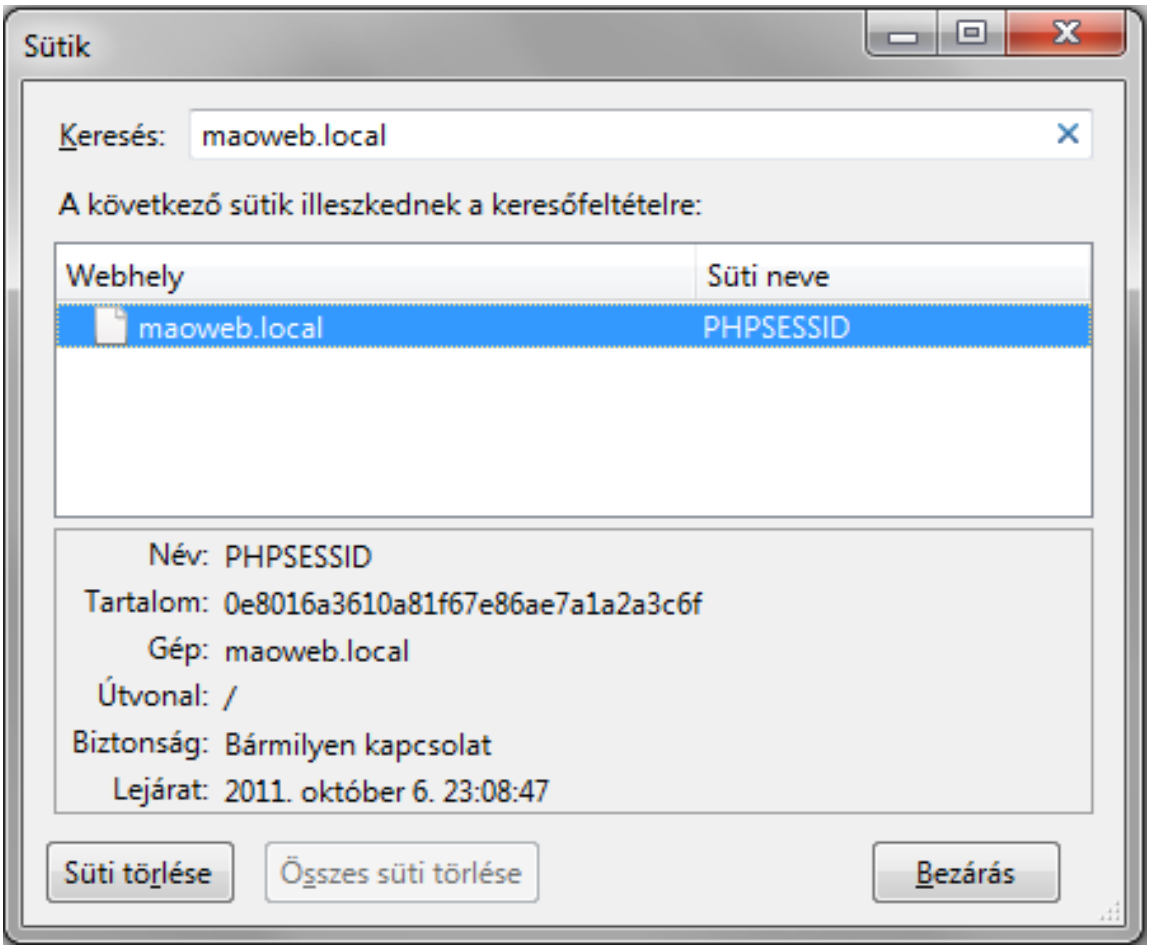
5. A programozás és tervezés során felmerülő problémák

Ebben a részben mint ahogyan azt pár sorral feljebb is említettem a programozás és tervezés során elém gördülő akadályokról fogok beszélni. Arról, hogy milyen nehézségekkel kellett szembenéznem és hogyan orvosoltam ezeket, részletesen az alpontokban fogok beszámolni.

5.1 A beléptető rendszer, avagy a session kezelés

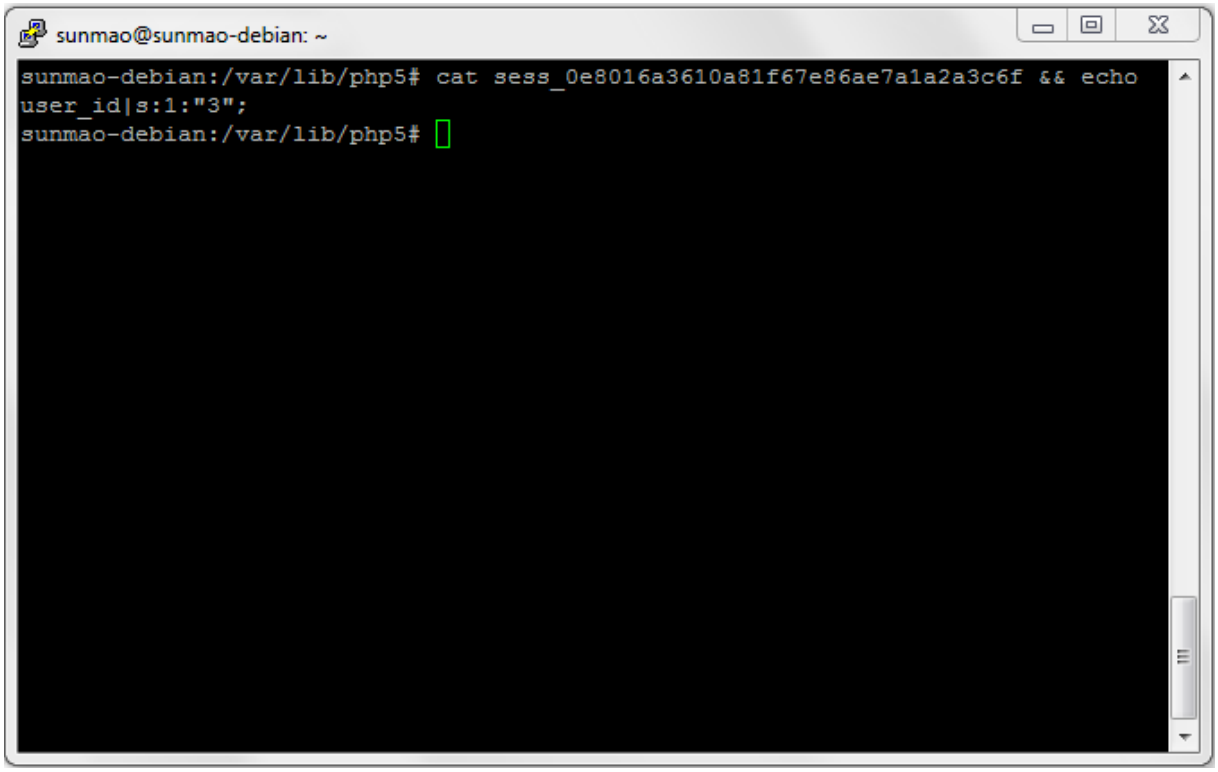
Mivel egy olyan programról van szó, ahol az adatokat valamilyen szinten védeni kell, ezért úgy döntöttem, hogy készítek egy beléptető rendszert a programhoz. Az első verzióban ez egy egyszerű session kezelés volt, aztán később ez egy igen összetett rendszerré fejlődte ki magát, de ne haladjunk ennyire előre, először is mi az a session és mire jó. A session szót magyarra fordítva munkamenetet kapunk. Mikor használunk sessionöket? Ha két számítógép kommunikál egymással és az egyik, de akár mindkét gépen is átmenetileg adatokat szeretnénk tárolni. A munkameneteket leginkább a böngésző és a webszerver közötti kapcsolat kibővítésére használják. Ehhez használnak még egy speciális azonosítót is ami nem más mint a session cookie. A folyamat a következőképpen néz ki. Mikor először meglátogatunk egy weboldalt a webszerver küld nekünk egy azonosítót amivel majd később igazolhatjuk magunkat. Ezt az azonosítót a böngésző lokálisan az úgynevezett cookie-ba (magyarul süti) tárolja.

Ezzel párhuzamosan a webszerver letárolja az adatbázisában az adott azonosítóhoz tartozó értékeket. Esetünkben, hogy be van-e jelentkezve a felhasználó vagy nincs, ha be van, akkor a felhasználóhoz tartozó user_id-t, ha nincs akkor semmit. Minden egyes oldal lekérésnél a böngésző elküldi ezt a sütiben tárolt azonosítót a webszervernek, amely ezáltal eldönti, hogy az adott személynek milyen jogosultságai vannak, illetve mik a vele kapcsolatos adatok. Ha például van egy bejelentkező felületünk, ahol megadjuk a helyes név, jelszó párost, akkor a rendszer elmenti az azonosítónkhoz kapcsolódva azt, hogy mi most beléptünk és ezután tovább enged minket a védett oldalakra. Ezek után ha a védett oldalon belül újabb hivatkozásra kattintunk, már nem kell még egyszer megadnunk a nevet és a jelszót, mert a rendszer tudni fogja, hogy ezzel az azonosítóval már sikeresen bejelentkeztek.



7. ábra: A kliens oldalon tárolt süti

A képen láthatjuk, hogy a süti neve **PHPSESSID** és tartalma a **0e8016a3610a81f67e86ae7a1a2a3c6f** karakterekből álló azonosító. Beállíthatjuk még emellett, hogy meddig tárolja a számítógépünk az adott sütit. Erről később majd részletesebben is lesz szó, ez az érték most 2011.október 6.



8. ábra: Az adott session-höz tartozó adatok a szerver oldalon

Ezen az ábrán SSH-n keresztül csatlakozunk a webszerverhez és láthatjuk, hogy az adatok egy **sess_0e8016a3610a81f67e86ae7a1a2a3c6f** nevű fájlban vannak eltárolva. A fájl neve mindig a sess-szóval kezdődik, és ehhez fűzi hozzá a rendszer az adott azonosítót. A fájl tartalma kezdetben üres, majd ha bejelentkezünk, akkor mint azt fentebb is láthatjuk a user_id lesz a tartalma. Esetünkben most a hármas user_id-jü felhasználóval vagyunk bejelentkezve. Gondolhatnánk, hogy ezzel a bejelentkezés rész meg is van oldva. Ám sajnos ez korántsem igaz, mivel ahány webszerver annyiféle beállítás. Mindenki tapasztalta már azt, ha más nem, a Neptun felületén, hogy egy bizonyos idejű tétlenség után a rendszer kidob minket és újra be kell jelentkeznünk. Ezzel ellentétben én azt szerettem volna, hogy például két óra tétlenség után is ott tudjuk folytatni a munkánkat ahol abbahagytuk.

Ehhez utána kellett járnom annak, hogy hogyan is működik ez az egész session kezelés. Az Apache webszerverek beállításától függően a következőképpen néz ki a sessionök kezelése. Alapértelmezetten a szerver végez egy úgynevezett szemétgyűjtést, aminek az a lényege, hogy a régebbi fájlokat kitörli az átmeneti könyvtárból. Ha mi semmilyen tevékenységet nem végzünk, akkor a szerveren tárolt **sess_azonosító** fájlunk elévül, és egy idő után mikor a szemétgyűjtő végig megy a fájlokon kitörli azt is. Ez viszont tudjuk, hogy azzal jár, hogy az adott kliens többé nem tudja magát azonosítani. Ennek következtében a rendszer kidob minket. Ezen problémára több megoldás is létezik, én először arra gondoltam, hogy a konfigurációs fájlban ezt a törlési időközt beállítom egy kellően nagy értékre és akkor sokáig megmaradnak azok a fájlok ahol az adatok vannak tárolva. Mivel én egy hordozható és minél megbízhatóbb rendszert szerettem volna összeállítani mégsem ez bizonyult a legjobb megoldásnak. Végül is úgy döntöttem, hogy az egyes session-ökhöz tartozó beállításokat nem egy könyvtárban fogom letárolni, hanem azokat is MySQL táblákban. Most már értjük, hogy miért hoztuk létre a fentebb látható manage_session nevű táblát. Most a következőképpen néz ki a bejelentkezés menete. Először is az elküldött session azonosítót megpróbálja megkeresni a rendszer a saját könyvtárában, ha itt nem találja, akkor a MySQL adatbázishoz fordul és itt kezd el keresni. Ha itt sincs olyan sor amely illeszkedik a megadott azonosítóra, akkor a felhasználó valóban nincs bejelentkezve. Ellenben ha talál illeszkedést, akkor a felhasználó be van jelentkezve, tehát folytathatja a munkát a védett részen belül. Emellett még azt is megcsinálja nekünk az ellenőrző rendszer, hogy ha valóban volt illeszkedés, akkor létrehozza a könyvtárában a megfelelő fájlt és beleírja a felhasználó user_id-jét, ezzel jelezve, hogy ő be van jelentkezve. Így most egy meghatározott ideig megint nem lesz szükség a MySQL tábla lékérdezésesre.

5.2 Kompatibilitási problémák

Sajnos a webfejlesztésnél sosem lehetünk igazán biztosak a dolgunkban. Ennek főképp az az oka, hogy nincs egy tökélesetesen egységes szabvány ami mindent lefed. Ahány böngésző annyiféle implementáció. Természetesen léteznek törekvések amelyekkel próbálják standardizálni a meglévő implementációkat, de ez igen nehéz feladat. Mind emellett jelen van még az is, hogy a felhasználóknak különféle beállításaik vannak. Vegyük a legegyszerűbbet a felbontást. Ez ugye monitor és videokártya függő és tudjuk, hogy ezekből nagyon sokféle van.

Így ha egy oldalt úgy optimalizálunk, hogy az 1024*768-as felbontásban jól nézzen ki, akkor lehet 800*600-ban már szét fog csúszni. Persze erre azért már a CSS elég jó megoldásokat nyújt, de vegyük akkor a telepített programokat. Van ahol nincsen telepítve például flash player, ezeknél a személyeknél a csak flash-el készült oldalak máris „kiestek”. Van olyan felhasználó aki biztonsági vagy egyéb okok miatt kikapcsolja a JavaScript támogatást. Tehát láthatjuk, hogy nagyon sok befolyásoló tényező lehet, ami a programozó dolgát nagyban megnehezíti. Velem is számtalanszor megesett már, hogy otthon minden működött tökéletesen, aztán egy másik böngészőbe megnéztem ugyanazt az oldalt és szörnyen nézett ki, illetve a funkciók sem működtek. Ebben az utolsó fejezetben azt fogom bemutatni, hogy én személy szerint milyen problémákkal találkoztam. Azt még mindenképpen érdemes megemlíteni, hogy a HTML5-el nagyon sok ilyen jellegű probléma meg fog oldódni, de ennek használata még nem lehetséges, mivel nem támogatja elég böngésző, illetve még nem elég elterjedt.

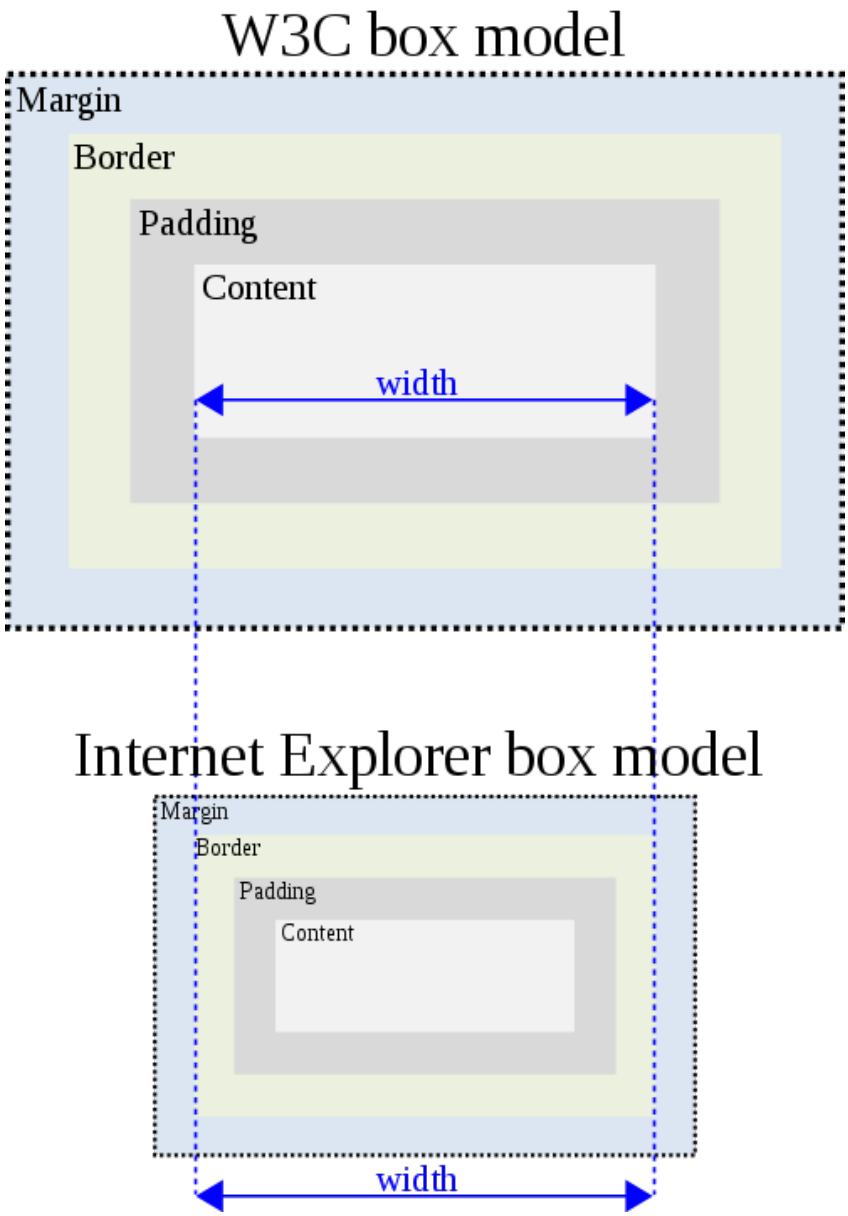
5.2.1 CSS

Amikor bemutattam a felhasznált programnyelveket, akkor már beszéltem a CSS-ről. Arról azonban nem, hogy vannak részek amit például az Internet Explorer és a Mozilla Firefox teljesen másképpen értelmez. Egy ilyen problémába futottam bele jómagam is, amikor a „beállítások” menüpontot terveztem. A CSS-ben létezik egy úgynevezett box model vagy magyarul doboz modell, amely a következő képen néz ki.

9. ábra: A doboz modell felépítése

Tárgyaljuk ki egy kicsit részletesebben, mit is látunk a képen. Az világos, hogy a „Hello World” az a tartalom lesz ami a HTML oldalunkon megjelenik. Kifelé haladva a belső margóval (padding) tudjuk megadni, hogy mekkora legyen a távolság a doboz széle és a benne lévő tartalom között. Esetünkben a „Hello World” szöveg és a doboz széle között. Azt is megmondhatjuk, hogy nézzen ki a doboz széle azaz a szegély (border). Milyen legyen a stílusa, vastagsága, színe, ez alapértelmezetten egyébként nem látszik. A külső margó (margin) lesz az egész doboz távolsága többi tartalomtól. Ha megértettük és átláttuk a doboz modellt, akkor nagyon elegáns weboldalakat tudunk készíteni viszonylag kevés munkával. Mint azt már említettem a problémába akkor futottam bele, amikor ugyanazt az oldalt megnyitva a két böngészőben más-más eredményt kaptam. Sokáig gondolkodtam azon, hogy ennek mi lehet az oka. Míg végül hosszas keresgélés után rájöttem, hogy az IE másképpen értelmezi a doboz modellt, de lássuk most már konkrétan, hogy mi is volt a probléma és annak megoldása. A szabvány szerint a szélességhez (width) nem tartozik hozzá a belső margó, a szegély és a külső margó. Ezzel szemben az IE beleszámolja ezeket is a teljes szélességbe, ezért ha például megadunk szélességnek 200 pixelt, illetve padding-nak és border-nek 20-20 pixelt, akkor IE-ben az fog történni, hogy összenyomódik a tartalom 120 pixelre és a doboz mérete lesz 200 pixel.

Normális esetben azt várnánk, hogy az egész tartalom marad 200 pixel és a doboz mérete nő meg 280 pixelre.



10. ábra: A kétfajta értelmezési mód

Most, hogy már tudjuk mi a probléma, nézzük mi lehet rá a megoldás. A helyzet az, hogy nem csak egy megoldás létezik az adott problémára. Az első lehetőségünk, az hogy használjuk a megfelelő doctype-ot, amellyel szabványos módba kapcsoljuk az értelmezőt.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Erről már beszéltem fentebb, ez is azon próbálkozások közzé tartozik amelyeknek célja az egységesítés. Egy másik megoldás lehet, ha úgy ügyeskedünk, hogy két külön divet hozunk létre, és az egyiknél beállítjuk a szélességet (width) a másiknál pedig a belső margót és a szegélyt. Lehetőségünk van még arra is, hogy úgy kommentezzük fel a HTML oldalunkat, hogy ha IE-vel nyitjuk meg az oldalt, akkor egy másik CSS fájlt használjon. Ennek az az előnye, hogy tisztább, átláthatóbb lesz a kód, viszont több CSS fájlt kell karbantartanunk.

5.2.2 JavaScript

Sajnos a JavaScript is azon nyelvek közzé tartozik melynek használata kockázatos. A kockázatost olyan szempontból értem, hogy sosem lehetünk biztosak benne, hogy ami az egyik gépen egy adott böngészővel működött, az egy másik gépen egy másik böngészővel is ugyanazt az eredményt produkálja majd. A legtöbb böngészőben ki lehet kapcsolni a JavaScript támogatást és emellett léteznek olyan böngészők is amelyekben nincs is ilyen támogatás. Persze ez ma már nagyon ritka és inkább akkor lehetséges ez, ha például Linux alatt nincsen grafikus felületünk és valamilyen karakteres böngészőt szeretnénk használni, mondjuk links-et. Ha ez a helyzet akkor egy olyan oldalt amelynek funkciói csak a JS-re hagyatkoznak nem tudunk használni. Azért az esetek nagy részében azt állíthatjuk, hogy a JavaScript használata jó ötlet. Ez kicsit furcsán hangozhat miután elsoroltam, hogy milyen problémák lehetnek ezzel a programnyelvvel, de mégis a legtöbb felhasználónál ez a funkció be van kapcsolva, mivel ha nem lenne, az oldalak nagy részét vagy teljes mértékben nem tudnák használni, vagy ha tudnák is akkor sem érnék el nagyon sok funkcióját. Ha azt vesszük alapul, hogy mennyi nagy oldal használ JS-t – Facebook, Google, Gmail és még sokan mások – akkor szerintem jogosan gondolhatjuk azt, hogy ez nem fog problémát jelenti. A kompatibilitási problémákat a legegyszerűbben úgy lehet megoldani, ha minél kevésbé speciális kódot írunk. Próbáljunk mindig a legegyszerűbb és legstandardabb módon programozni. Ami még hasznos lehet a segédkönyvtárak használata, viszont abból is olyat kell választanunk amely sok böngészőt támogat. Most szeretnék bemutatni egy olyan kompatibilitási problémát amit csak részben tudtam megoldani.

A célom az volt, hogy a valamilyen tulajdonság alapján kilistázott hallgatókat ki lehessen nyomtatni. Ezt a következő képen oldottam meg. A táblázat paneljéhez hozzáadtam egy gombot, ami megnyit egy külön ablakot, csak a táblázat adataival, a nyomtatáshoz megfelelően méretezve. Majd ezen az oldalon meghívtam a **print()** függvényt. Érdekes módon Firefox és Internet Explorer alatt tökéletesen működött a nyomtatás funkció, de a Google Chrome nevű böngészője már nem vette az akadályt. Aztán fórumok hosszas olvasgatása után kiderült, hogy a **print()** függvény gyorsabban lefut mint ahogyan az új ablak megnyílik ezért kell bele némi késleltetést tenni. Nevezhetnénk ezt programozói hibának is, de akkor a fentebb említett másik két böngésző alatt miért működött kifogástalanul a dolog? Mindenesetre most úgy működik a nyomtatás funkció, hogy a program eldönti, hogy milyen böngészőt használ az adott személy és ennek függvényében késlelteti a nyomtatást vagy egyből meghívja a **print()** függvényt.

6. Összegzés

A kitűzött célt sikerült megvalósítanom, ami az volt, hogy egy olyan programot hozzak létre, amely segítségével viszonylag könnyen – komolyabb informatikai ismeretek nélkül – lehet szakdolgozatot készítő hallgatókat adatbázisba menteni. Összességében azt tudom elmondani a programról, hogy közel sem tökéletes, de véleményem szerint nem létezik tökéletesen elkészített program. Arra viszont mindig törekedni kell, hogy a lehető legjobbat hozzuk ki a programunkból. Értem ezt mind a hatékonyságra, mind a stabilitásra és a biztonságra. Én úgy érzem, ezen program elkészítésével elindultam egy úton, ami mindenképpen pozitív irányba halad. A tervezés és megvalósítás során nagyon sokat tanultam mások hibáiból, de még többet a sajátjaimból. Amellett, hogy bővültek a programozási ismereteim, lehetőségem volt a Linux operációs rendszert, illetve azon belül a szerver oldali szolgáltatásokat is egy kicsit jobban megismerni. A programozást ezek után sem szeretném abbahagyni, sőt szeretném továbbfejleszteni magam és új dolgokat megvalósítani.

7. Irodalomjegyzék

E. Gamma, R. Helm, R. Johnson, J. Vlissides: Programtervezési minták, Kiskapu kiadó, 2004

http://en.wikipedia.org/wiki/Model–View–Controller

http://en.wikipedia.org/wiki/Apache_HTTP_Server

http://hu.wikipedia.org/wiki/JavaScript_library

http://hu.wikipedia.org/wiki/MySQL

http://hu.php.net/manual/en/pdo.setattribute.php

http://modzone.web4.hu/cikk/az-egyke-singleton-tervezesi-minta

http://www.tizag.com/phpT/phpsessions.php

8. Külső forrásból származó ábrák

2. ábra:

http://flipsidereality.com/blog/wp-content/uploads/rtorrent_imgs/22.png

3. ábra:

http://upload.wikimedia.org/wikipedia/hu/e/e4/PhpMyAdmin_kepmentes.png

4. ábra:

http://www.ghacks.net/wp-content/uploads/2009/12/mysql_workbench_main.png

6. ábra:

http://www.bhartisoftland.com/technologies-skill-sets/gifs/mvc-php.png

9. ábra:

http://xhtml.com/C12742E4-3FD2-4084-9E7F-833805A4157C/box-model.gif

10. ábra:

http://upload.wikimedia.org/wikipedia/commons/thumb/6/64/W3C_and_Internet_Explorer_box_models.svg/500px-W3C_and_Internet_Explorer_box_models.svg.png