SZAKDOLGOZAT

Gacsal Patrik

Debrecen
2009

Debreceni Egyetem
Informatikai Kar

PORTÁLMOTOR FEJLESZTÉSE

Témavezető:
Dr. Kuki Attila
Egyetemi adjunktus

Készítette:
Gacsal Patrik
Mérnök informatikus

Debrecen
2009

Tartalomjegyzék

1. Bevezetés

Úgy érzem bátran kijelenthetem, hogy az internet napjaink elsődleges információforrásává vált. Segítségével bárki publikálhat, bármilyen témában és minőségben, számos formában, mint például cikk, blog, fórum, stb. Az ilyen módon publikált hol hasznos, hol teljesen haszontalan információkra könnyen rálelhetünk az információ típusától függően ilyen-olyan keresőmotorok segítségével.

Ez a fajta információközlés általában úgynevezett weblapokon keresztül történik, olyan formában, hogy a felhasználó elindít egy böngészőt, beír egy URL (Uniform Resource Locator, azaz egységes erőforrás-azonosító) címet annak címsorába, majd a böngésző megjeleníti a címhez tartozó lapot. Ezen lapok alapvetően két osztályba sorolhatók; lehetnek dinamikus, vagy statikus tartalmúak. A statikus tartalmú lapok segítségével csak egyirányú kommunikáció valósítható meg, viszont egyszerű az elkészítésük, de nagyobb, összetettebb oldalaknál nehézkessé válhat kezelésük. A dinamikus tartalmú oldalak ezzel szemben lehetővé teszik a többirányú kommunikációt, számos fejlett szolgáltatással együtt. Ezek elkészítése és karbantartása viszont nem egyszerű feladat, elengedhetetlen hozzá néhány programozási és leíró nyelv ismerete.

A CMS (Content Management System, azaz tartalomkezelő-rendszer) rendszerek hivatottak azt a célt szolgálni, hogy bárki, akár programozói tudás nélkül is létre tudjon hozni és a tartalmát kezelni egy saját dinamikus tartalmú oldalnak, vagy immár portálnak. Emiatt ezeket a rendszereket szokás még portálmotoroknak is nevezni, habár ez az elnevezés távolról sem hivatalos és egy kicsit homályos is. Rengeteg, több ezer ilyen rendszer létezik már, némelyikük ingyenes és nyílt forrású, némelyikük nem, vannak általános célúak és vannak bizonyos célterületekre specializálódottak. Néhány a legismertebb, ingyenes, általános célú CMS-ek közül: Joomla!, Drupal, E107, PHPnuke, PHPFusion.

Az előző bekezdésben sikerült érintenem egy fogalmat, amely nevezetesen a "portál" volt. A portál definíciója szerint egy weboldal, amely különböző forrásokból származó adatokat jelenít meg egységesített formában és az egyszerű keresőmotortól kezdve mindenféle szolgáltatást nyújt látogatóinak, mint például e-mail, hírek, készletinformáció, árak, egyéb információ, szórakoztatás, stb. A portálok alapvetően két csoportra oszthatók. Az egyik csoportba tartoznak a horizontális portálok, amelyek ugyan sok témával foglalkoznak, de

többnyire csak érintőlegesen. A másik csoport képviselői a vertikális portálok, amelyek kevesebb témával foglalkoznak, de komolyabb mélységekben. Ezeken belül számos osztálya van még a portáloknak: személyes, regionális, hivatali, céges, sport, stb.

A téma kiválasztásban rengeteget segített az a tény, hogy régóta szeretnék egy igazán sokoldalú, teljesen egyedi, saját portált építeni. Ha egy létező CMS segítségével tenném, akkor nem lehetne igazán egyedi, ha pedig csak egyszerűen megírnám az új portál kódját az adott feladatra specializálva, akkor problémás lenne a kód újrahasznosítása. Ebből meg is született az ötlet: saját CMS-t kell készítenem.

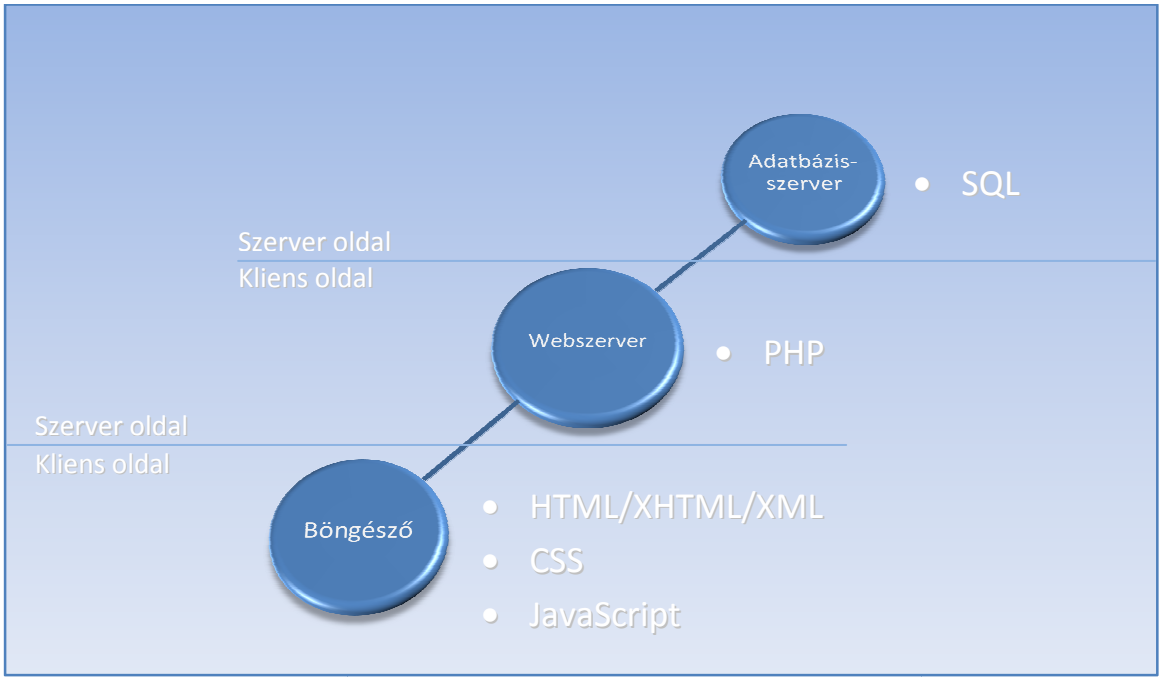
A célom egy működőképes, lehető legszélesebb körben alkalmazható, moduláris felépítésű CMS, vagyis portálmotor megalkotása, amely egyfajta keretrendszerként szolgálna további fejlesztéseimhez. A rendszer ergonómiája ebben a formában csak másodlagos, a lényeg, hogy működjön.

A készülő rendszerrel szemben támasztott elvárásaim következők:

- moduláris, modulorientált felépítés
- tartalomkezelés, szerkezeti módosítások lehetőségének biztosítása
- megjelenítendő oldalak elrendezésének részletes testreszabhatósága, lehetővé téve bármilyen úgynevezett template, más néven weblapsablon alkalmazását
- többnyelvűség támogatása
- felszínváltoztatás támogatása
- eltérő jogkörök, jogosultsági szintek támogatása
- ingyenes webhosting, azaz tárhely szolgáltatók által biztosított környezetben is legyen működőképes
- a lehető legtöbb fajta böngészőben adja ugyanazt az eredményt.

2. Alkalmazott nyelvek és kapcsolatuk

A böngészőnkben megjelenő lapok többnyire HTML (HyperText Markup Language, azaz hiperszöveges jelölőnyelv) nyelvű dokumentumok, amelyek átvitele HTTP (HyperText Transfer Protocol), és HTTPS (Hypertext Transfer Protocol Secure) protokollok segítségével történik. Ezen protokollok kliens-szerver architektúrára épülnek, a kapcsolódás egy irányba történhet, csak a kliensek felől és csak a szerver felé, így megkülönböztethető egy kliens és egy szerver oldal.



2.1. Kliens oldal

A kliens oldalon a kliens, vagyis a böngésző által végrehajtható műveletek halmazát értjük. Ez egy kicsit leegyszerűsítve annyiból áll, hogy le tudja kérni és meg tudja jeleníteni a HTML dokumentumot, ha pedig nem HTML, akkor továbbadja a megfelelő alkalmazásnak. A HTML dokumentum tartalmazhat beágyazott scripteket, stíluslapokat, ilyen-olyan objektumokat. Természetesen a böngészőnek ezeket is interpretálnia kell valahogyan. Például egy Adobe flash objektumot az Adobe által fejlesztett plugin segítségével, egy JavaScript kódot pedig saját értelmezőjével futtat.

2.1.1. HTML (HyperText Markup Language)

A HTML egy strukturált dokumentum leíró nyelv, amelyet weboldalak készítéséhez fejlesztettek ki. A HTML specifikációját W3C (World Wide Web Consortium) felügyeli. Most a HTML 5-ödik (2008) változata az aktuális, de ezt még nem mindegyik mai böngésző támogatja. A HTML előző, 4.01-es (1999) változatához képest számos, többek között multimédiával kapcsolatos újítása van. Jóval a HTML 5 előtt jelent meg az XHTML (2000), amely XML alapokra helyezte a HTML-t. A végeredmény egy kicsit szigorúbb szintaktikával rendelkező HTML 4.01 lett. Azóta megjelent az XHTML 5 is, amely a HTML 5 XML változata. Az említett nyelvek mind az SGML (Standard Generalized Markup Language, azaz szabványos általános jelölőnyelv) egy konkrét alkalmazásai.

Az SGML nyelvekben a dokumentum struktúráját és kinézetét speciális szimbólumok, úgynevezett "tag"-ek segítségével lehet kialakítani. A tag-eket mindig < és > jelek határolják, és általában van belőle egy nyitó és egy záró tag is. A záró tag annyiban különbözik a nyitótól, hogy kiegészül egy / jellel az elején, például `<p>ez egy HTML bekezdés</p>`. Az XHTML megköveteli a záró tag nélküli tag-ek bezárását is, abban a formában, hogy a tag neve után hozzáfűzünk egy szóköz + / jelet, így: `<br />, <img src='kép.jpg' />`.

Egy HTML dokumentum három fő részre osztható:

1. Dokumentum Típus Definíció (DTD), a dokumentumnak megfelelő HTML/XHTML/XML típust és verziót rögzíti. Ez mindig az első sorban kell, hogy legyen. Például:
   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
   ```

2. HTML fejrész, a dokumentumra vonatkozó meta adatokat, keretek információit, címet, karakterkódolást tartalmazza. Mindig `<head></head>` tag-ek határolják.

3. HTML törzs, amely tartalmazza a dokumentum leírását, ez kerül a `<body></body>` tag-ek közé.

Egy példa:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
   <title>Az oldal címe</title>
   <meta http-equiv="Content-type" content="text/html;charset=iso-8859-2">
 </head>
 <body>
  <h1>Ez egy címsor</h1>
  <p>ez egy bekezdés</p>
  <!-- Ez pedig egy megjegyzés, amely csak a forrásban látható -->
 </body>
</html>
```

A rendszer által generált dokumentumok a HTML 4.01 ajánlásának felelnek meg, azért választottam ezt a dokumentum típust, mert ez a legelterjedtebb, egyszerűen nincs böngésző, amely ne tudná megjeleníteni.

2.1.2. CSS (Cascade Style Sheets)

A CSS egy stílusleíró nyelv, amely a HTML, XHTML és bármilyen fajta XML dokumentum stílusának leírására használható. Stíluslapok már az SGML megjelenésével is léteztek 1970 elején, de a CSS első változata mégis csak 1996 végén került ajánlásra a W3C által. Az aktuális változat a CSS 2, amely 1998-ban jelent meg. A CSS 3 azóta is, napjainkig fejlesztés alatt áll. A HTML 4-edik változatával megszületett egy új irányelv, miszerint a stílusleíró elemeket el kell különíteni a tartalomtól. A CSS használatának rengeteg előnye van még azon kívül, hogy jobban áttekinthető a vele készített dokumentum. Például sokkal részletesebben be lehet állítani az egyes elemek megjelenítését, akár egyenként, akár csoportosan. Gyorsítótárazni is lehet, mivel a tartalomra nincs hatással csak a megjelenítésére, így dinamikus oldalaknál sem okoz problémát.

Egy stíluslap szabályok listáját tartalmazza. Minden szabály egy szelektorból és egy deklarációblokkból áll. A szelektor határozza meg, hogy mely elemre/elemekre lesz érvényes az őt követő deklarációblokk. A szelektorban hivatkozhatunk elemtípusra (a, li, td, table, stb), osztályra (.osztálynév), azonosítóra (#azonosító) és pszeudo-osztályra (:pszeudo-osztály), vagy akár minden elemre (*), és mindezek leszármazottjaira a dokumentumfában (szóköz). A deklarációblokk deklarációk listáját tartalmazza, amelyek egyszerű tulajdonság-érték párosok.

Például:

```
p #nagy_voros {      /* szelektor és deklarációblokk kezdete*/
 font-size: 18pt;    /* deklaráció */
 color: #550000;     /* deklaráció */
 line-height: 1.5em; /* deklaráció */
}                    /* deklarációblokk vége */
/* minden <p>bekezdés</p> nagy_voros azonosítóval rendelkező eleme 18
pontos vörös betűkkel lesz kiírva és a sormagasság 1.5 sor lesz */
```

Stíluslapokat rendelni egy dokumentumhoz lehet beágyazással, vagy csatolással. Az előbbi megoldás úgy történik, hogy `<style>` és `</style>` tag-ek közé helyezzük a szabálylistát a dokumentumban. Stíluslap csatolásakor a `<link>` tag segítségével adhatjuk meg a csatolni kívánt CSS fájlt a dokumentum fejlécében, vagy `<style src='CSS_fájl.css'></style>` formában a dokumentumtörzsben.

2.1.3. JavaScript

A JavaScript egy objektum-orientált scriptnyelv. Nincs igazán rokoni kapcsolatban a Sun Microsystems által fejlesztett Java nyelvvel, nevét mindössze a két nyelv szintaxisának hasonlósága miatt kapta. Elsődleges felhasználási területe a kliens oldali programozás, vagyis aktív tartalom létrehozása a böngészőkben. Manapság minden komolyabb oldalon előfordul kisebb-nagyobb mértékben. Minden böngésző támogatja, viszont az interpreter implementációk böngészőnként és akár verziónként is komoly mértékben különbözhetnek egymástól. Így könnyen meglehet, hogy egy kód tökéletesen fut a Firefox 3.x.x böngészőben, de a Firefox 2.x.x és az Internet Explorer már nem tud vele mit kezdeni.

A JavaScript mint nyelv nem típusos és annak ellenére, hogy szinte teljesen objektum alapú, csak nagyon limitált objektum-orientált eszközrendszerrel rendelkezik. Pontosabban nem is rendelkezik objektum-orientált eszközrendszerrel, csak azt részben szimuláló eszközökkel. A C által bevezetett vezérlési szerkezetek mind használhatók JavaScriptben is, szóval ez is egy úgynevezett C szintaxisú nyelv, mint a Java, PHP és még sokan mások.

Az interakció a JavaScript kód és a dokumentum között a DOM (Document Object Model) segítségével zajlik. A dokumentum bármely részét elérhetjük és módosíthatjuk kedvünk szerint, új oldalt tölthetünk be, űrlapokat küldhetünk el és még sorolhatnám.

Egy dokumentumban általában valamilyen eseményhez kötjük a scriptek futtatását. Ilyen esemény például az oldal betöltődése, klikkelés, duplaklikkelés, vagy csak az egérmutató mozgatása egy elemen. Minden HTML elem rendelkezik eseménykezelő attribútumokkal, mint például az `onclick`, `ondblclick`, `onmouseover`, `onmouseout`, `onmousedown`, `onmouseup`, stb.

Scriptet rendelni egy dokumentumhoz a stíluslapokhoz hasonlóan lehet csatolással, vagy beágyazással. A beágyazás `<script></script>` tag-ek segítségével történik a dokumentumtörzsben. A csatolás a `<link>` tag segítségével történik a dokumentum fejlécében, vagy a `<script>` tag `src` attribútumát használva a dokumentumtörzsben.

Egy példa:

```
<html>
 <body>
  <p onmouseover='fv();'>ez egy bekezdés</p>
  <!-- ha a fenti bekezdés felé kerül az egérmutató,
       akkor megjelenik egy 'Üzenet' egy szövegdobozban -->
  <script language='JavaScript'> <!-- script kezdete -->
   function fv() { // ez egy függvény deklaráció
    // meghívjuk a window DOM objektum alert() metódusát
    window.alert('Üzenet');
   }
  </script> <!-- script vége -->
 </body>
</html>
```

2.2. Szerver oldal

Jelen esetben a szerver oldalon a webkiszolgáló által végrehajtható művelek halmazát értjük. Statikus oldalak esetében itt mindössze annyi történik, hogy egy kérés érkezésekor a kiszolgáló egyszerűen továbbadja a kért oldalt, a saját tárolójából a látogató böngészője felé. Dinamikus oldalaknál már komplikáltabb a helyzet, mivel a tartalom számos változó függvényében változhat. Ilyenkor a kiszolgáló általában kiterjesztés alapján, egy előfeldolgozó egységnek adja át a kért oldal kódját, majd annak kimenetét továbbítja a böngésző felé, amely immár az által is értelmezhető HTML, XHTML, vagy XML nyelvű dokumentum, vagy bármi más. Több ilyen előfeldolgozó, vagy másik nevén szerver oldali script is létezik, például a PHP (PHP: Hypertext Preprocessor), ASP (Active Server Pages),

JSP (JavaServer Pages). Ezek közül a legelterjedtebb talán a PHP, mivel nagyon könnyű programozni, hihetetlenül sokrétű, ingyenes és ráadásul multi platformos is.

2.2.1. PHP (PHP: Hypertext Preprocessor)

A PHP elsődlegesen egy szerver oldali scriptnyelv, vannak egyéb felhasználási területei is, de ezeket csak ritkán használják, futtatható parancssorból, vagy akár komoly kliens oldali alkalmazások is készíthetők segítségével. A nyelv 1995-ben született meg, de mai formáját csak olyan 2000 körül érte el, a PHP negyedik verziójával. Itt már rendelkezett objektum-orientált eszközrendszerrel, habár ez még távolról sem volt teljes. Nem volt láthatóság kezelés, interfészek és az összetettebb, objektumokat tartalmazó kifejezéseket sem tudta kezelni. Ezen apró hiányosságok többek között mind pótolva lettek a PHP 2004-ben megjelent 5-ödik verziójában. A PHP nagyjából a C++ szintaxisát követi. Mint scriptnyelv, nincs szükség hozzá fordítóra, futási időben értelmezi egy úgynevezett interpreter. A PHP segítségével írható a HTTP fejléc, kapcsolat létesíthető adatbázis kiszolgálókkal, képek manipulálhatók, munkamenetek kezelhetők és még nagyon-nagyon sok hasznos szolgáltatást nyújt felhasználói számára.

A PHP interpreter kiterjesztés alapján kapja meg az értelmezendő fájlt a webkiszolgálótól. A kapott fájl kódjában `<?php` és `?>` tag-ek határolják a valóban PHP nyelven íródott kódrészleteket, ezt értelmezi, a többi egyenesen megy a kimenetre, azaz a böngésző felé. A HTTP fejléc írása csak addig történhet meg, amíg semmi sem került a kimenetre. Mivel a sütik is a HTTP fejlécben tárolódnak, ezek kezelését is ekkor kell megoldanunk. Ez a tény később fontos szerepet játszik majd az általam fejlesztett rendszer kialakítása során.

Az adatok küldése kliens oldalról a szerver oldal felé történhet GET és POST metódusok, valamint sütik használatával. Ez utóbbinak saját érvényességi ideje van, tehát nem csak egy oldaltöltés erejéig őrzi meg adatait, mint az előző kettő, hanem akár évekig is. A süti a felhasználó számítógépén tárolódik, a használt böngésző által meghatározott helyen és formában. A GET metódussal küldött adatok látszanak a böngésző címsorában is az oldal címét követő ? jel után, & jellel elválasztott név=érték párosokban, valahogy így: http://site.com/index.php?nev1=ertek1&nev2=ertek2. A POST metódussal küldött adatok nem látszanak sehol a böngészőben, általában űrlapok adatait szokták így küldeni. Ezeket az adatokat a HTTP titkosítás nélkül közvetíti, csak a HTTPS titkosítja.

Példa egy PHP kód beszúrására:

```
...
<?php
 // Ez már PHP kód
 // kiírunk egy szöveget sortöréssel a kimenetre
 echo "Helló világ!<br />";
?>
...
```

A rendszer programkódjai a PHP 4.1 szerinti specifikációra épülnek. A PHP 5 jóval fejlettebb, de számos tárhely szolgáltatónál 4.x.x van telepítve, így hozzájuk igazodva maradtam eme régebbi verziónál.

2.2.2. SQL (Structured Query Language)

Az SQL, ahogy neve is mutatja, egy strukturált lekérdező nyelv, amely segítségével relációs adatbázis-kezelő rendszerek programozhatók. Alapjait az Edgar F. Codd által, 1970-ben megfogalmazott relációs adatmodell nyújtja. A nyelv első hivatalos változata 1986-ban született meg (SQL86), amelyet az ANSI (American National Standards Institute) jegyzett be. Azóta több jelentős kiadás is létrejött, ezek az SQL89, SQL92 és SQL99 szabványok. A nyelv jelentősége leginkább abban rejlik, hogy egységes programozói felületet nyújt alkalmazások és felhasználók, leginkább adminisztrátorok számára, bármilyen azt implementáló relációs adatbázis-kezelő rendszerhez. Számos ilyen adatbázis-kezelő rendszer létezik, de sajnálatos módon az egyes implementációk között akadnak különbségek, amelyek többnyire a kezelt adattípusokra korlátozódnak. Néhány ezen rendszerek közül: Oracle, PostgreSQL, MySQL, Firebird.

Az SQL nyelv utasításait három alnyelvre szokás bontani, amelyek név szerint a DDL (Data Definition Language, adatdefiníciós nyelv), DML (Data Manipulation Language, adatmódosító nyelv) és a DCL (Data Control Language, adatvezérlő nyelv). Mint programozási nyelv, az SQL részben deklaratív, részben procedurális. Szintaxisát tekintve egyedi, az alkalmazási területére specializált, viszonylag egyszerű nyelvről beszélhetünk.

Egy példa a MySQL használatára PHP környezetben:

```
<?php

/* példa kód, amelyben kapcsolódunk egy MySQL adatbázishoz,
   küldünk hozzá egy SQL kérést,
   majd az eredményt kiírjuk egy HTML táblázatba. */


$kapcsolat = mysql_connect("kiszolgáló_URL", "felhasználónév", "jelszó");
$kérés = "SELECT * FROM tábla";
$eredmény = mysql_query($kérés);

$i = 0;
$mezőnevek = array();
while (false !== $mezőnév = @mysql_field_name($eredmény, $i++))
 $mezőnevek[] = $mezőnév;

echo "<table>";
echo "<tr>";
foreach ($mezőnevek as $mezőnév)
 echo "<th>$mezőnév</th>";
echo "</tr>";

while (false !== $sor = mysql_fetch_array($eredmény)) {
 echo "<tr>";
 foreach ($mezőnevek as $mezőnév)
  echo "<td>" . $sor[$mezőnév] . "</td>";
 echo "</tr>";
}
echo "</table>";

mysql_free_result($eredmény);
mysql_close($kapcsolat);

?>
```

A rendszer MySQL 4 adatbázis kiszolgálókkal való együttműködésre lesz felkészítve, mert a legtöbb tárhely szolgáltató is ezt biztosítja.

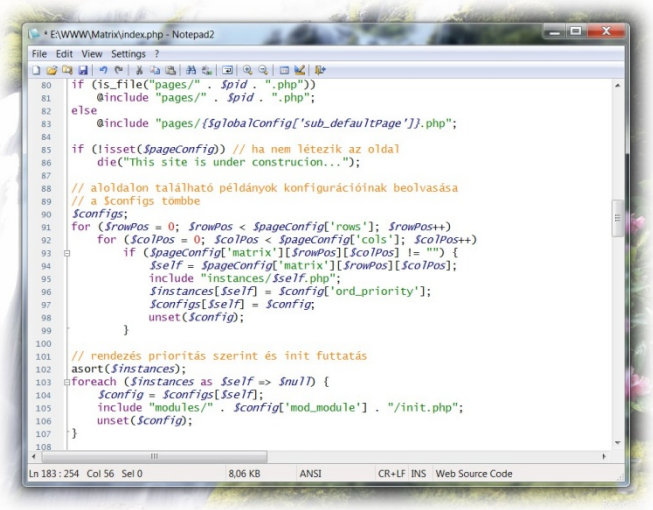
Maga a MySQL egy többfelhasználós, többszálú, relációs adatbázis-kezelő kiszolgáló. A szoftvert a svéd MySQL AB fejlesztette, amely kettős licencezéssel tette elérhetővé azt. Választható módon vagy a GPL, vagy egy kereskedelmi licenc érvényes a felhasználásra. 2008-ban a Sun Microsystems felvásárolta a céget, amelyet 2009-ben felvásárolt az Oracle Corporation.

A szoftver első változata 1995-ben jelent meg, azóta az 5.1.40 jelölésű stabil verziónál jár. Természetesen multiplatformos, amely vélhetőleg nem kis részben járult hozzá elterjedéséhez. Napjainkban több, mint 6 millió telepített példánya van használatban világszerte.

3. Fejlesztői eszközök, környezet

Az összes előző részben tárgyalt nyelven, összefoglaló nevükön scriptnyelveken történő fejlesztésre tökéletesen alkalmas egy egyszerű Windows jegyzettömb, de ez valójában a legminimalistább megoldás. Léteznek nagyon komoly IDE (Integrated Development Environment, Integrált fejlesztői környezet) rendszerek, amelyek épp csak nem írják meg helyettünk a programunk kódját. Ilyenek például a Dreamweaver és a NetBeans is. Ezek a programok minden fejlesztői környezettel szembeni elvárásnak megfelelnek. Kódszerkesztőjük nem csak kiemeli a nyelv szintaxisának megfelelően a beírt kód szavait, hanem vizsgálja is, hogy nem-e vétettünk valami hibát gépeléskor, így arra már akkor fény derül. Segítenek a kód írásában is, mindenféle név- és formális-paraméterlista kiegészítő funkcióval, gyorsbillentyűkkel és makrókkal, hogy minél kevesebbszer kelljen ugyanazt begépelnünk. Mindezeken felül fejlett stílusszerkesztő is társul az említett eszközök mellé, sok más hasznos kiegészítő funkció között.

Én a Florian Balmer által fejlesztett Notepad2 Kai Liu jóvoltából továbbfejlesztett változatát választottam munkám programozási részének elkészítéséhez. Ez egy kiterjesztett funkcionalitású jegyzettömb, többnyelvű szintaxis kiemeléssel és bővített szerkesztési lehetőségekkel. A fentebb említett fejlesztői környezetek létezésének tudatában ez a választás értelemszerűtlennek tűnhet, viszont én már jó ideje ezt használom és nagyon megkedveltem.

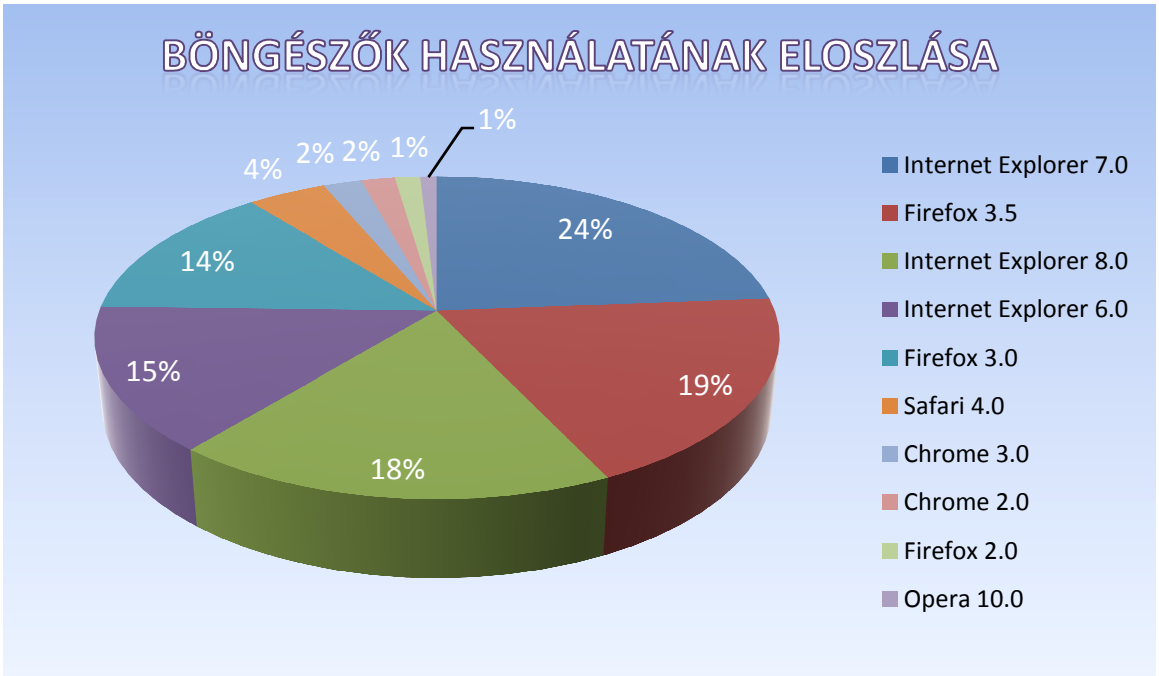


a Notepad2

A grafikai feladatok elvégzéséhez az Adobe Photoshop Creative Suite 3 képszerkesztő programot választottam, habár ezek a feladatok nem voltak sokan, és nem is túl jelentősek. Sok segítséget nyújtott a stílusdefiníciók írásakor, ezen belül a színek megadásakor a Sam Francke által fejlesztett, CPick nevű kis programocska.

A szerver oldali futtatási környezet a Romain Bourdon által készített, WAMP (Windows Apache MySQL PHP) névre keresztelt programcsomag által lett biztosítva. A csomag neve sokat elárul tartalmáról, az első betű rögzíti a platformot, így van LAMP, MAMP, SAMP és FAMP is, a többi betű a csomagban található kiszolgáló alkalmazásokra utal.

Kliens oldali futtatási környezetként használtam a Microsoft Internet Explorer 6, 7 és 8-as változatát, és a Mozilla Firefox 2.x.x, 3.0.x, és 3.5.x verzióit. Véleményem szerint, ha valami mindezeken jól működik, akkor más böngészővel is mennie kell. A Firefox 2.x.x és főleg az Internet Explorer 6 sok kellemetlen meglepetéssel tudnak szolgálni, igaz lassan történelemmé válnak, de sajnos még mindig vannak olyanok, akik használják őket.



forrás: w3counter.com/globalstats.php, 2009. szeptember

4. A Mátrix

A mátrix a matematikában számok téglalap alapú elrendezése (táblázata), pontosabban olyan absztrakt mennyiségek táblázata, melyeket összeadni és szorozni lehet. Mátrixokat szoktak használni lineáris egyenletek leírására, lineáris transzformációk együtthatóinak és olyan adatok tárolására, melyek két paramétertől függenek.

A fejezet címe igencsak sejtelmesre sikeredett, de most már elárulom, hogy készülő rendszerem kapta ezt a nevet, tervezett működése alapján. Egy jó névválasztás már fél siker, mivel elsődlegesen ez különbözteti meg egymástól az egyes termékeket. Egy jó névnek rövidnek, könnyen megjegyezhetőnek, vagy valahonnan ismerősnek, fülbemászónak kell lennie. Az általam választott Mátrix név már igencsak bejáratott, a matematika és a film- és játékipar által is. De mivel csak egy egyszerű kísérleti rendszerről van szó, mindez teljesen lényegtelen, habár talán egyszer felnő majd a nevéhez.

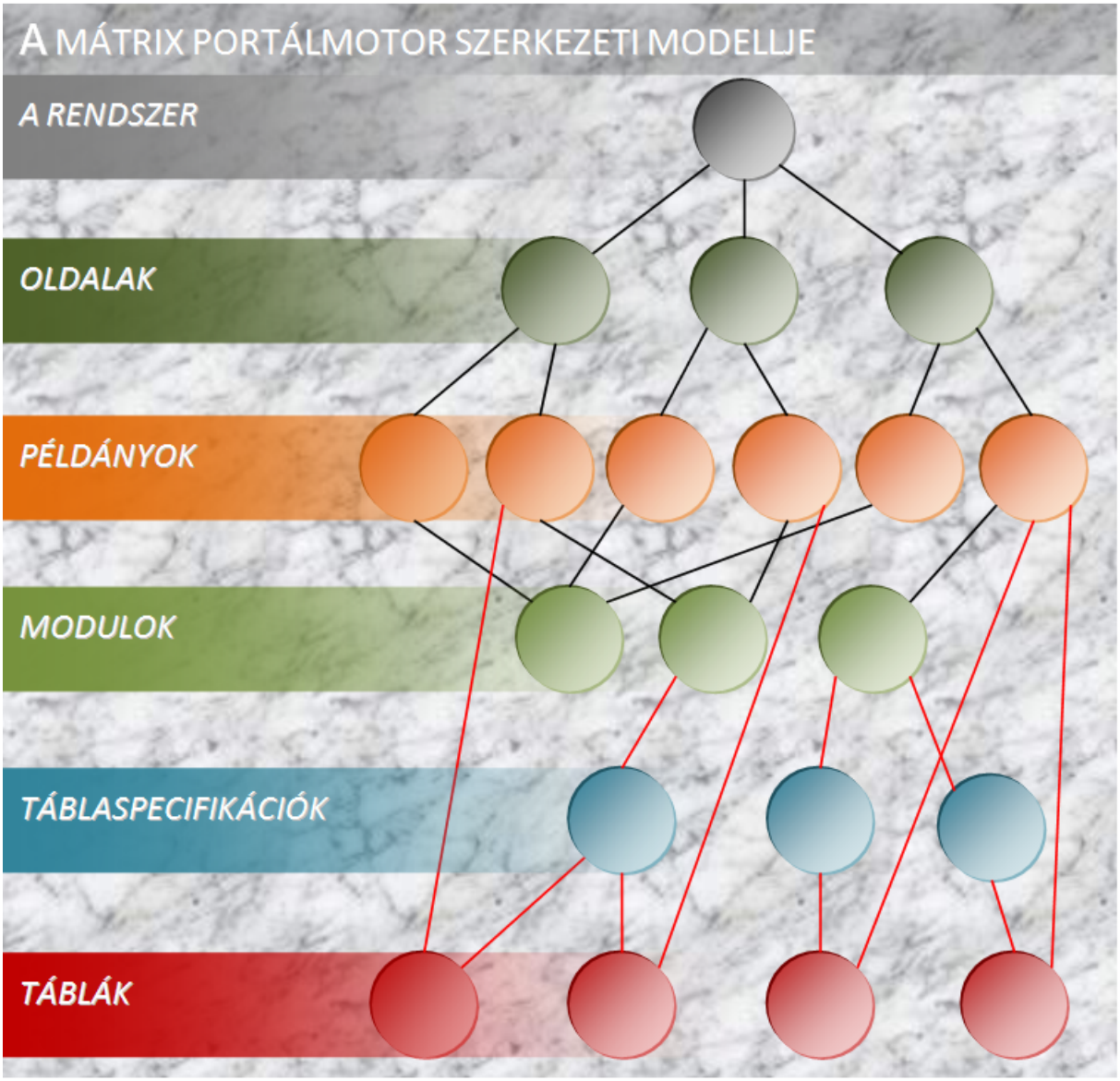
4.1. Alapkoncepció

Weblapok böngészése, vizsgálata és készítése közben azt az érdekes megfigyelést tettem, hogy túlnyomó többségükre ráilleszthető egy celláit megfelelően egyesített táblázat olyan módon, hogy az egyes cellái a lap logikailag elkülönülő egységeit tartalmazzák. Még nem találkoztam olyan statikus elrendezésű weblappal, amelyre ez az elmélet ne lett volna igaz.

Az említett logikailag elkülönülő egységek, amelyeket nevezzünk egyszerűen blokknak, általánosságban rendelkeznek pár figyelemreméltó tulajdonsággal:

- Feladatuk többnyire egyszerű, mint például kiolvasnak valamit egy adatbázisból majd megjelenítik, vagy éppen rögzítenek valamilyen adatokat.
- Programkódjuk sokszor nagyon hasonló, például fórum - blog és hírek - üzenőfal. Ezeket szinte csupán a bevitel módja különbözteti meg egymástól.
- Bizonyos erőforrásokat megosztva is használhatnak (fájlok, adatbázis táblák), például amikor megjelenik a nap képe egy képgaléria által tartalmazott képek közül, naponta véletlenszerűen kiválasztva.

Ezen megállapítások alapján meg is született a rendszer modellje, három entitásra és egy erőforrástípusra, valamint egy azt meghatározó interfészre alapozva. A rendszer entitásai név szerint az oldal, példány és a modul. Az erőforrástípus a tábla, és az azt meghatározó interfész pedig a táblaspecifikáció.



Az oldal definiálja a megjelenítendő lap elrendezését szabályozó táblázatot annak sor- és oszlopegyesítéseivel együtt, a celláiban elhelyezkedő példányokat, valamint egyéb adatokat.

A példány saját névvel és konfigurációval rendelkező, egy tetszőleges modul kódjára támaszkodó entitás.

A modul egy paraméterezhető programkód, amely egy bizonyos feladat végrehajtására van megírva, de paraméterezhetőségének köszönhetően, annak mértéke függvényében felhasználási területe és kinézete változhat.

A rendszerben használt adatbázis táblák szerkezetét a modulok határozzák meg táblaspecifikációik által, a köztük fennálló kapcsolatokat pedig a példányok. Egy adott példányhoz csak olyan táblákat lehet hozzárendelni, amely megfelel a hozzá tartozó modul táblaspecifikációinak. Ezzel a rendszer adatbázis struktúrája csak részben rögzített és teljes egészében opcionális.

A modell létrehozásával meg is lett alapozva a rendszer elvi működése, nincs más hátra, mint a megvalósítás.

4.1.1. Oldal

A oldal egy érdekes szó, ugyanis szokás használni két értelemben is, jelentheti oldalak egy webkiszolgáló bizonyos mappájában található halmazát és azok egy konkrét elemét is. Az oldal, mint rendszerentitás, az utóbbi értelmezés szerinti jelentéshez kapcsolódik.

Az oldal rendszerentitás, továbbiakban csak egyszerűen oldal, feladata mindössze annyi, hogy leírja valamilyen formában a megjelenítendő oldalt, valamint annak néhány tulajdonságát, mint például a cím és egyéb, csak az adott oldalhoz kapcsolódó dolgot. A megjelenítendő oldal tulajdonképpen egy explicit módon megadott méretekkel és cellaegyesítésekkel rendelkező táblázat lesz, celláiban egy-egy példánnyal azonosítójával.

Az oldal fizikailag egy egyszerű PHP fájl, amelyben a $pageConfig asszociatív tömb kap értékeket. Így csak be kell fűzni a feldolgozó kódba (`include "oldal_azonosító.php";`), és már rendelkezésre is áll a $pageConfig tömb minden adattal, amelyre szükség lehet az oldal kirajzolásához. A PHP által kezelt tömbök lehetnek heterogén szerkezetűek is, ezáltal nem okoz problémát, ha az oldal minden tulajdonságát tekintet nélkül a típusára egy tömbbe sűrítjük. A $pageConfig tömbnek tartalmaznia kell a táblázat sorainak és oszlopainak számát, az egyes cellák sor- és oszlopegyesítéseit, tartalmát, színét (szerkesztéskor van jelentősége), valamint az oldal címét, leírását és a megtekintéshez szükséges jogosultsági szintet.

Példa egy ilyen oldalkonfigurációs fájlra:

```
<?php
$pageConfig['title'] = "kezdőlap";
$pageConfig['description'] = "alapértelmezett aloldal";
$pageConfig['rows'] = "5";
$pageConfig['cols'] = "2";
$pageConfig['authLevel'] = "0";
$pageConfig['matrix'][0][0] = "login";
$pageConfig['colspans'][0][0] = "1";
$pageConfig['rowspans'][0][0] = "1";
$pageConfig['colors'][0][0] = "#FFCC99";
$pageConfig['widths'][0] = "20%";
$pageConfig['matrix'][0][1] = "home";
$pageConfig['colspans'][0][1] = "1";
$pageConfig['rowspans'][0][1] = "3";
$pageConfig['colors'][0][1] = "#FFFFCC";
$pageConfig['widths'][1] = "80%";
$pageConfig['heights'][0] = "";
$pageConfig['matrix'][1][0] = "menu";
...
?>
```

4.1.2. Példány

A modulok példányosítása kézenfekvő megoldásnak tűnt ezen feladat megvalósítására. Egy modul példányait saját konfigurációjuk, pontosabban azonosítójuk, alapvető és modul specifikus tulajdonságaik különböztetik meg egymástól. Ezek a példányok nemes egyszerűséggel a példány nevet viselik, hasonlóan az oldalhoz. A megvalósítás szintén az oldaléra emlékeztető. A példányok konfigurációját egy egyszerű PHP állomány tárolja, amely definiálja a $config asszociatív tömböt. A fájl neve a példány azonosítója és a ".php" kiterjesztés konkatenációja.

A példányok tulajdonságai két csoportra oszthatók, vannak alapvető és modul specifikus tulajdonságok. A példányok alapvető tulajdonságait a "base" modul definiálja. Ezek a tulajdonságok minden példánynál jelen vannak, egyikük határozza meg azt, hogy melyik modul példánya az adott példány. Ebben a formában ráfogható, hogy a modulok között létezik egyfajta kétszintű hierarchikus öröklődési viszony, azaz mindegyik modul a base modul leszármazottja. A base modul példányai az üres példányok. A modul specifikus tulajdonságokat az egyes modulok specifikálják.

A példányok tulajdonságai típusosak, amelyre a név négykarakteres prefixuma utal. A típusok az tulajdonság érték-hozzárendelésekor játszanak fontos szerepet, egyrészt megakadályozzák

a helytelen konfigurálást, másrészt megkönnyítik azt, egyes típusok esetében a lehetséges értékek felajánlásával.

Egy példa egy példány konfigurációs fájljára:

```
<?php
// alapvető tulajdonságok
$config['mod_module'] = "login";
$config['bln_windowed'] = true;
$config['bln_uniqueStyle'] = false;
$config['str_title'] = "bejelentkezés";
$config['str_icon'] = "";
$config['ord_priority'] = "0";
$config['sty_containerStyle'] = "";
$config['bln_isVisible'] = true;
$config['bln_stopAutoRowspan'] = false;
$config['str_text'] = "";
// modul specifikus tulajdonságok
$config['ord_timeout'] = "1000";
$config['bln_changeSID'] = true;
$config['bln_oneRow'] = false;
$config['bln_compact'] = true;
$config['sub_registration'] = "reg";
$config['sub_lostPass'] = "";
$config['sub_modUserData'] = "mod";
$config['tbl_users'] = "users";
$config['tbl_activeUsers'] = "ausers";
$config['tbl_avatar'] = "avatars";
?>
```

4.1.3. Modul

A modul a legösszetettebb és a legfontosabb jelenség az egész rendszerben. A moduloknak a fentebb említettekkel ellentétben aktív programkódjuk van, amelyek összehangolt működése adja magát a portált. Ezek a programkódok paraméterezhetőek, így konfigurációtól függően másképp viselkedhetnek. A modulok konfigurációit a példányok képviselik. A modulok programkódjai két csoportra bonthatók, inicializáló és fő programkódra. Az inicializáló kód fut le először, amely futtatási sorrendjét az oldalon található többi példányhoz képest egy alapvető paraméter, a prioritás határozza meg. Az inicializáló kódok nem adhatnak semmilyen kimenetet, így innen írható a HTTP fejléc is, amelynek többek között átirányításkor, süti írásakor van jelentősége. A fő programkódok futtatása az oldalt reprezentáló táblázatban elfoglalt cella szerint, helyhez kötötten történik. A fő programkódoknak már lehet kimenetük és általában van is, mivel ezekből tevődik össze az oldal. A modulok része továbbá egy

példányok felé interfészt biztosító speciális konfigurációs fájl, amely definiálja a modul paraméterneveit, azok típusait és egyes típusú paraméterek specifikációit.

A modul fizikailag egy modul nevével megegyező nevű mappa és annak tartalma a rendszerben. E mappa tartalma feltételszerűen a következő fájlokat tartalmazza:

- args.php - paramétereket leíró fájl, interfész
- init.php - inicializáló programkód
- main.php - fő programkód
- egyéb, saját függvényeket és osztályokat tartalmazó fájlok

Az args.php definiálja az $args tömböt, valamint opcionálisan a $tableSpecs és az $options asszociatív tömböket. A $args tömb elemei a paraméterek nevei, amelyek négykarakteres prefixuma azok típusára utal. Pontosabban a négy karakterből három, a negyedik csak elválasztó szerepet tölt be ('_'). A típusok az érték-hozzárendelésnél játszanak fontos szerepet. Ezek a típusok a következők:

- bln - logikai
- ord - egész, vagy lebegőpontos szám
- str - rövid szöveg
- txt - hosszú szöveg
- sty - stílus leírás
- tbl - tábla, szerkezetét a $tableSpecs tömb írja le
- sub - oldal
- mod - modul
- ins - példány
- sel - választható értékű, értékkészletét az $options tömb tartalmazza

A $tableSpecs tömb tábla típusú paraméterek nevével megegyező kulcsú elemei olyan tömbök, amelyek az adott tábla mezőneveit és azok SQL nyelven értelmezhető definícióit tartalmazza. Később ez alapján lehet majd táblákat létrehozni, illetve hozzárendelni egy példányhoz. Például:

```
$args[] = "tbl_tabla";
$tableSpecs['tbl_tabla'] =
     array("mezo1", "INT(6) NOT NULL AUTO_INCREMENT", "mezo2", "INT(3)");
```

Az $options tömb "sel" típusú paraméterek nevével egyező kulcsú elemeket tartalmaz, amelyek szintén tömbök, és az adott paraméterhez tartozó értékkészletet tárolják.

Például:

```
$args[] = "sel_valaszthato_erteku_parameter";
$options['sel_valaszthato_erteku_parameter'] =
         array("érték1", "érték2", "érték3", "érték_n");
```

Az init.php és main.php fájlok programkódjai futtatáskor olyan módon rendelődnek a példányokhoz, hogy előbb a példány, majd a modul fájljai kerülnek befűzésre, így a programkódok futásakor már elérhető a $config asszociatív tömb, amelynek kulcsai az $args tömb elemei. Például:

```
include "peldanyok/peldany.php";
// itt már jelen van a $config tömb
include "modulok/modul/init.php";
```

Mivel egy modul kódja egy PHP futási ciklusban többször is befűzésre kerülhet, így az azokon belüli konstans, függvény és osztály deklarációkat a program kódján kívül, más fájlokban kell elhelyezni és azokat include_once vagy require_once nyelvi elemekkel kell befűzni. Ezzel elkerülhető ezen nevek újradefiniálása, amely különben egy futtatás leállítását eredményező, kritikus hibát generálna. A modulok kódjaiban számos változó és osztály elérhető, a keretrendszer szolgáltatásaként. A modulok kódjaira van néhány szabály is, amelyek figyelmen kívül hagyása esetén a rendszer instabillá válhat. Ilyen szabályok például, hogy a használt GET és POST változóneveknek legyen kezdőszelete a példány neve ($self), ne használjuk a rendszer változóit (lista a dokumentációban), deklarációkat külön fájlban tegyük meg és window.onload helyett használjuk a rendszer $onloadScripts tömbjét.

A legegyszerűbb, statikus tartalom megjelenítésére szánt modul (static) main.php kódja:

```
<?php

if (!defined("inMatrix"))
      die();

echo($config['txt_content']);

?>
```

4.2. Frontend felület

Általában minden CMS rendszernek van egy első és egy hátsó bejárata, vége. Ezeket szokás frontend illetve backend felületeknek nevezni. A frontend felület az, amelyet minden felhasználó lát és használ, amikor a portál oldalait böngészi. A backend felületet jó esetben csak a portál adminisztrátorai látják, akik ezen keresztül módosíthatják a portál szerkezetét, kinézetét és tartalmát. A Mátrix portálmotor is egy ilyen rendszer, mivel máshogy egyszerűen elképzelhetetlen a működtetése.

A frontend felület feladata mindössze az oldal legenerálása, a backend felületen keresztül létrehozott konfigurációs állományok alapján. Ezen feladat magába foglalja azt is, hogy a modulok programkódjai számára meg kell teremteni egy megfelelő futtatási környezetet. Ez a környezet tulajdonképpen néhány globálisan elérhető változóból áll. Ezek a következők lesznek:

- $authLevel - jogosultsági szint, értéke 0 és 3 között változhat, ezek a szintek a következők:
    - 0: vendég
    - 1: felhasználó
    - 2: kiemelt felhasználó
    - 3: adminisztrátor
- $user - felhasználó neve
- $uid - felhasználó azonosítója
- $params - ParamList objektum, amely az oldalkérés GET paramétereit tárolja és szelektíven lekérdezhető
- $onloadScripts - window.onload eseményhez kötött JavaScript függvények neveit tárolja.
- $globalConfig - asszociatív tömb, a rendszer globális beállításait tárolja
- $db - Database objektum, a kapcsolódó adatbázis kezelésében segít
- $pid - a kért oldal azonosítója
- $pageConfig - a kért oldal konfigurációja
- $configs - az oldalon található példányok konfigurációit tároló tömb
- $config - az aktuális példány konfigurációja

- $self - aktuális példány neve

Ezeken felül elérhető még egy inMatrix nevezetű konstans és a classes.php összes osztálya. Az inMatrix konstans a rendszer alapszintű védelmére szolgál abban a formában, hogy megléte nélkül nem lehet a modulok programkódjait futtatni. A classes.php a rendszerhez szorosan kapcsolódó feladatok végrehajtására tervezett osztályok definícióit tartalmazza.

A globális változók minden modulkód számára írhatók és olvashatók, így megvalósítható a példányok interakciója más példányokkal, az aktuális oldallal és a rendszerrel is. Sajnos sok minden más is megoldható ezen tág határokon belül, amely előnyösnek nem mondható. Egy rosszul megírt modulkód futtatásakor könnyen széteshet az egész rendszer. De mivel ez egy teljesen moduláris, modulorientált rendszer, ez még éppen belefér a keretbe.

A frontend felület alapjai ezzel kész is volnának, már csak le kell programozni olyan formában, hogy a sok kis részlet egy egészet alkosson. Természetesen ez PHP nyelven kell, hogy megtörténjen.

Első lépésben meg kell akadályoznunk, hogy a kimentre bármi is kerüljön az inicializáló kódok futtatása befejezéséig. Ez azért lehet fontos, mert a HTTP fejléc csak addig írható, amíg nem került semmi a kimenetre. Ha pedig véletlenül egy apró hiba miatt mégis úgy sikerülne, akkor az további hibákat eredményezhetne. Szóval mentsük ami menthető alapon, ártani biztosan nem árthat. A PHP lehetőséget nyújt a kimenet pufferelésére, ezt fogjuk most kihasználni.

```
ob_start();
```

A következő lépés az inMatrix konstans definiálása.

```
 define("inMatrix",  true);
```

Ennek csupán annyi értelme van, hogy az ezután befűzött kódokban definiálva lesz ez a konstans. Ez php-n kívülről nem lehetséges, így a következő sort az első sorban tartalmazó programkódok önállóan nem lesznek futtathatók:

```
if (!defined("inMatrix")) die();
```

Be kell fűzni a classes.php-t, azaz a rendszer osztályainak definícióit tartalmazó fájlt, hogy azok a későbbiekben elérhetőek legyenek.

```
include "classes.php";
```

Kezdőértékekkel kell ellátni a környezet változóit, mindenképp explicit módon, mert a PHP bizonyos direktíva bekapcsolása mellett képes a kívülről érkező változókat saját nevükkel elérhetővé tenni a programkódban (register_globals).

```
$authLevel = 0;
unset($user, $uid);
$params = new ParamList();
$onloadScripts = array();
$configs = array();
```

A rendszer konfigurációs fájljait is be kell fűzni, ha léteznek. Ilyenek az adatbázis-kapcsolatot (dbcfg.php) és a rendszer konfigurációját (cfg.php) leíró fájlok. Ezeket a fájlokat a backend felületen lehet létrehozni, módosítani. A globális konfigurációs fájl megléte követelmény a továbbjutáshoz, ha nem létezik, akkor a program megáll egy üzenettel. Az adatbázis opcionális, elméletileg anélkül is működőképes a frontend felület, de a backend nem, így csak szélsőséges esetekben képzelhető el ez a helyzet.

```
if (is_file("cfg/cfg.php"))
      include "cfg/cfg.php";
else die("This site is under construction...");

if (is_file("cfg/dbcfg.php")) {
      include "cfg/dbcfg.php";
      $db = new Database($host, $dbase, $user, $pwd);
      unset($host, $user, $pwd, $dbase);
}
```

Meghatározzuk a kért oldal azonosítóját és be is fűzzük. Ezzel létrejön a $pageConfig tömb, amely az oldalt leíró adatokat tartalmazza.

```
if (isset($_GET['pid']))
      if (is_file("pages/" . $_GET['pid'] . ".php"))
            $pid = $_GET['pid'];
      else
            $pid = $globalConfig['sub_defaultPage'];
else
      $pid = $globalConfig['sub_defaultPage'];
```

```
include "pages/$pid.php";
```

Ezután össze kell gyűjteni a $pageConfig tömbből az oldalon található példányokat az $instances tömbbe és azok konfigurációit a $configs tömbbe. Az $instances tömb kulcsai az oldalon található példányok nevei lesznek, a hozzájuk tartozó érték az adott példány prioritása. A $configs tömb kulcsai szintén a példánynevek lesznek, a hozzátartozó értékek pedig a példányokhoz tartozó $config tömbök.

```
$instances = array();
for ($rowPos = 0; $rowPos < $pageConfig['rows']; $rowPos++)
      for ($colPos = 0; $colPos < $pageConfig['cols']; $colPos++)
            if ($pageConfig['matrix'][$rowPos][$colPos] != "") {
                  $instance = $pageConfig['matrix'][$rowPos][$colPos];
                  include "instances/$instance.php";
                  $instances[$instance] = $config['ord_priority'];
                  $configs[$instance] = $config;
                  unset($config);
            }
```

Az $instances tömbre csak azért van szükség, hogy prioritás szerinti sorrendbe rendezhessük a példányokat. Ez azért szükséges, mert az egyes példányok építhetnek egymás eredményeire. Például a bejelentkeztető (login) modul többek között megváltoztatja az $authLevel, vagyis a jogosultsági szint értékét, bizonyos adatok alapján. A modulok kódjai egyenként futnak le. Ha ez a bejelentkeztető modul kódja fut utoljára, akkor az összes többi abban a tévhitben fog élni, hogy a látogató egy egyszerű vendég, vagyis nincs bejelentkezve. Akkor ezt a rendezést most meg is tesszük és lefuttatjuk a példányokhoz tartozó inicializáló kódokat:

```
asort($instances);
foreach ($instances as $self => $null) {
      $config = $configs[$self];
      include "modules/" . $config['mod_module'] . "/init.php";
      unset($config);
}
```

Mostanra meg kellett, hogy történjen a bejelentkeztetés, megvizsgálhatjuk, hogy van-e jogunk az oldal megtekintéséhez. Ha véletlenül nem lenne, akkor átirányítjuk a böngészőt a megfelelő hibaoldalra:

```
if ($pageConfig['authLevel'] > $authLevel)
     header("Location:?pid={$globalConfig['sub_authorizationErrorPage']}");
```

Ezen a ponton befejeztük a HTTP fejléc írását, megkezdhető a kimenet írása. Leállítjuk a kimenet pufferelését és kiürítjük a puffert:

```
ob_end_flush();
```

Most már kiírható a Dokumentum Típus Definíció (DTD) és a dokumentum fejléce is:

```
echo <<<END
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset={$globalConfig['sel_charsetEncoding']}">
<link rel="StyleSheet" href="themes/{$globalConfig['sel_theme']}.css"
type="text/css">
<title>{$globalConfig['str_title']} {$pageConfig['title']}</title>
</head>
END;
```

A soron következő kiírandó rész a dokumentum törzse. Itt fog létrejönni az oldal konfigurációja által leírt táblázat. A kirajzolás a tradicionális két egymásba ágyazott for ciklussal valósul meg. A külső ciklusmag csupán a táblázat sorainak nyitását és zárását végzi. A belső ciklusmag rajzolja ki a táblázat celláit és azokon belül futnak le a modulok fő programkódjai, létrehozva az oldal tartalmát. Ezen kódok futtatását a $self változó értéke alapján konkrétan a base modul végzi, amely eközben értelmezi az adott példány alapvető paramétereit és annak megfelelően jár el.

```
... // az áttekinthetőségért a megjelenítésre vonatkozó részeket kihagytam
echo "<table>";
for ($rowPos = 0; $rowPos < $pageConfig['rows']; $rowPos++) {
      echo "<tr>";
      for ($colPos = 0; $colPos < $pageConfig['cols']; $colPos++) {
            if ($self = $pageConfig['matrix'][$rowPos][$colPos] != "") {
                  ...
                  echo "<td ...>";
                  $config = $configs[$self];
                  ...
                  include "modules/base/main.php";
                  unset($config, $self);
                  echo "</td>";
            } else
                  echo "<td ...></td>";
      }
      echo "</tr>";
}
echo "</table>";
```

Mindezek után már csak az $onloadScripts által tartalmazott, vagyis a window.onload (oldal betöltődése) eseményhez kötött JavaScript scriptek futtatását kell megoldani. Ennek jelentősége abban rejlik, hogy így bármelyik modulkód tartalmazhat ilyen eseményhez kötött scriptet anélkül, hogy azok akadályoznák egymást a futásban. A megvalósítás úgy fog kinézni, hogy létrehozunk egy JavaScript függvényt, amely sorban elindítja a tömb által tartalmazott függvénynevekhez kapcsolódó függvényeket, majd ezt a függvényt hozzárendeljük az említett eseményhez.

```
echo "<script language='JavaScript'>";
echo "function runGlobalScripts() {";
foreach ($onloadScripts as $script)
      echo "      $script();";
echo "}";
echo "window.onload = function() {runGlobalScripts();};";
echo "</script>";
```

A dokumentumtörzs, a dokumentum és a php kód lezárásával el is készült a frontend felület motorja. Itt említeném meg, hogy a beszúrt forráskód részletek az olvashatóság kedvéért többnyire leegyszerűsített változatok, csak nagyjából mutatják a motor működését.

```
echo "</body>";
echo "</html>";
?>
```

Ezzel úgy érzem sikerült létrehoznom azt a rendszert, amely bármilyen táblázatos szerkezetű oldalt képes megjeleníteni, tekintet nélkül annak tartalmára, mivel azt a példányok segítségével a modulkódok generálják. A frontend motor feladata mindössze a megjelenítés és a futtatási környezet megteremtése a modulok kódjai számára, amelyekre minden más, tartalomfüggő feladat elvégzése hárult. A modulok kódjainak megírása így valamivel nagyobb körültekintést igényel, mint más esetekben, de néhány egyszerű szabály betartásával megoldhatók.

4.3. Backend felület

A backend felület, hátsó bejárat célja a rendszer adminisztrálásának támogatása. Ezen a felületen keresztül módosítható a portál szerkezete, kinézete, tartalma esetlegesen más tulajdonságok mellett. Esetünkben a rendszer konfigurálását, entitásainak és erőforrásainak összehangolását kell lehetővé tennie. Ez a felület a frontenddel ellentétben nem igazán általánosítható, mivel előre meghatározott konkrét feladata van. Saját beléptető rendszerrel és a rendszert kezelni képes eszközökkel kell rendelkeznie. A backend felület nem feltétlenül webes, lehet egy önálló kliens alkalmazás is, de ebben az esetben az előbbiről lesz szó.

4.3.1. Első indítás

A backend, vagy ahogy az én rendszeremben elneveztem, adminisztrációs felület indítása úgy történik, hogy a böngésző címsorába beírjuk a megfelelő URL címet kiegészítve egy /admin szócskával. Első indításkor nem léteznek konfigurációs fájlok, nem hozható létre kapcsolat adatbázis-kiszolgálóval sem és így bejelentkezni sem lehet. Az adminisztrációs felület egy úgynevezett varázsló szerű megoldással segít ezen a problémán.

Az első lépésben megadható az adatbázis-kiszolgáló címe, adatbázis neve, a hozzá tartozó felhasználónév és jelszó. Ha a megadott adatok segítségével létrehozható a kapcsolat, akkor a konfiguráció mentésre kerül a rendszer cfg mappájában dbcfg.php néven. A fájl tartalma a kliens oldalról elérhetetlen, mivel egy olyan php kód, amelynek nincs semmilyen kimenete, így a letöltött fájl üres lesz. Ez igaz a rendszer többi konfigurációs fájljára is.

A második lépésben felvehető egy adminisztrátor a kapcsolódó adatbázis egy alapértelmezés szerinti, frissen létrehozott, felhasználókat tároló táblájába. Az adminisztrációs felület jogkörkezelése egyszintű, amely a frontend 3-as szintjének felel meg. Ezután már be lehet jelentkezni az itt megadott felhasználónév és jelszó segítségével.

Belépés után betöltődnek az alapértelmezett beállítások, így ez a felület működőképes, de a frontend mindaddig blokkolva lesz, amíg nem mentjük el manuálisan a beállításokat.

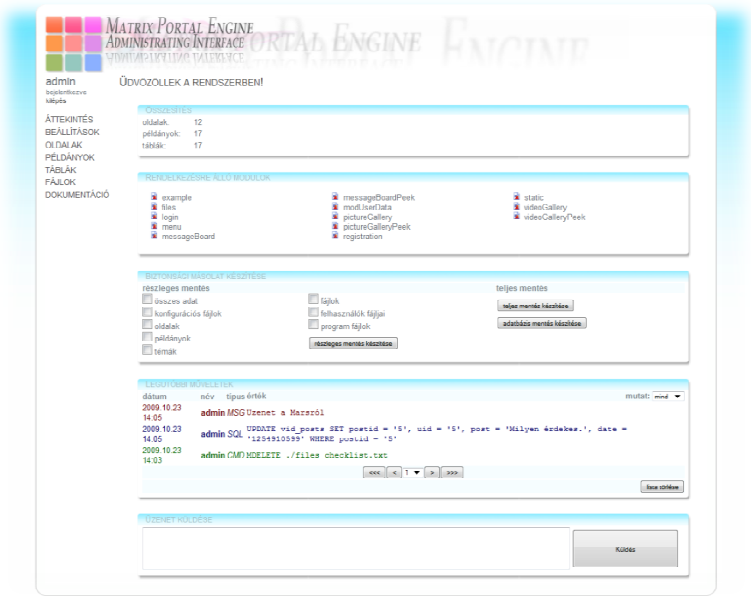
A felületet hét oldalra osztottam fel, amelyeket elsősorban egy navigációs menü kapcsol össze. Az elrendezés minden oldalon ugyanaz: felül van a címsor, alatta bal oldalon a felhasználó adatai és a menü, jobb oldalon az adott oldal címe és a hozzá tartozó tartalom. Az oldalak: áttekintés, beállítások, oldalak, példányok, táblák, fájlok, dokumentáció.

4.3.2. Áttekintés

Ezen az oldalon, amely egyébként a felület kezdőlapja, egy áttekintésszerű képet kapunk a rendszer aktuális állapotáról. Látható rajta a létrehozott oldalak, példányok és táblák száma, az elérhető modulok és a legutóbb végrehatott műveletek. Itt lehet eszközölni a biztonsági mentéseket, amely lehet részleges, teljes, fájlokra és adatbázisra vonatkozó is.

A végrehajtott műveletek listáján bejegyzések három típusa helyezkedik el, amelyekre szűkíteni is lehet a listázást. Ezek az SQL, CMD és MSG típusok. Az SQL az adatbázis felé küldött SQL kéréseket takarja. A CMD az SQL analógiájára általam létrehozott, a rendszerhez igazított egyszerű parancsnyelv. Segítségével fájlműveleteket lehet végezni ellenőrzött módon, egyszerű szintaxisú parancsokkal. Az ilyen parancsok sokkal inkább olvashatóak, könnyebben létrehozhatóak egy JavaScript kóddal, mint php megfelelőjük, különösen ha az ellenőrzéseket bele vesszük. Tehát a CMD típushoz tartozó értékek ilyen CMD parancsok. Az MSG egyszerű üzenetet takar, ilyen üzeneteket a lista alatt található szövegmezőbe írva és azt elküldve hagyhatunk. Ennek jelentősége több adminisztrátor léte esetén bontakozik ki igazán. Az egyes típusú bejegyzések más színnel jelennek meg.

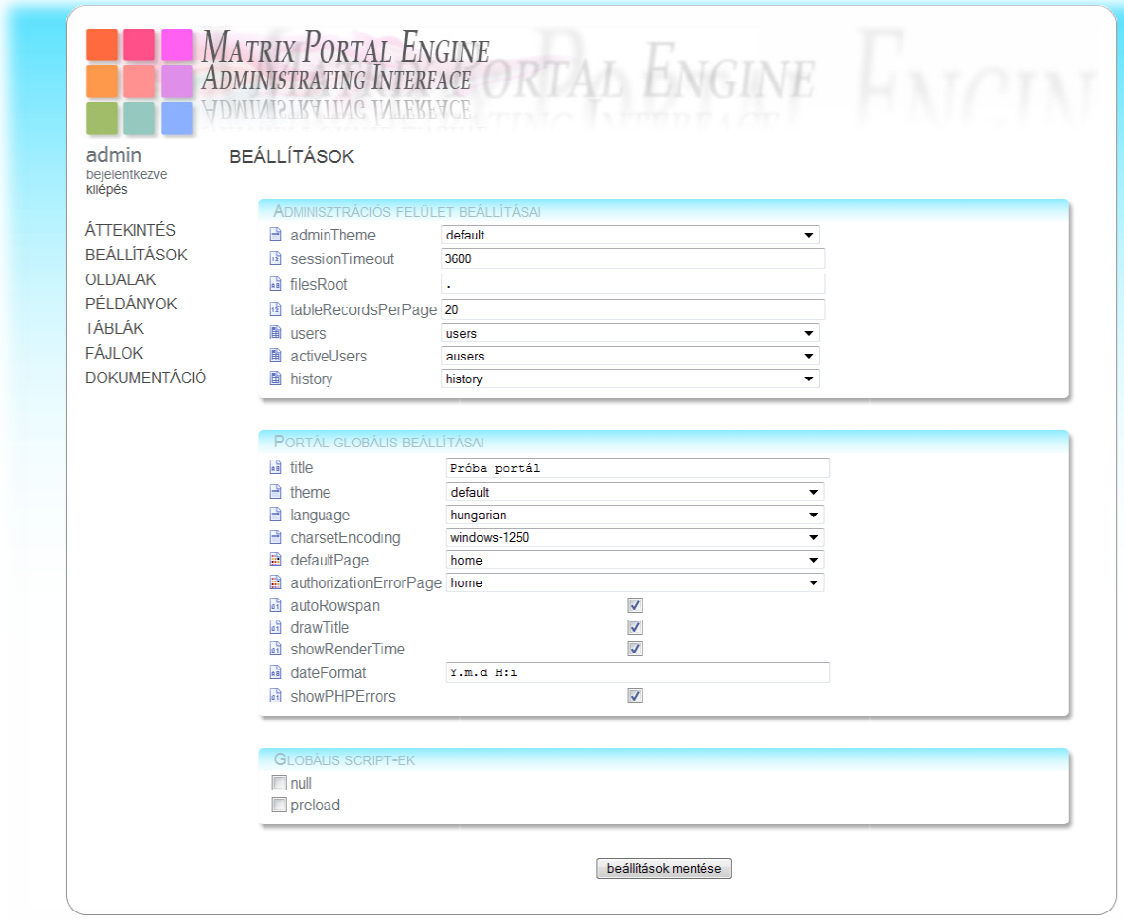
A biztonsági mentés fájlok esetén zip, adatbázisnál sql formátumban, csak letöltés formájában működik, a szerveren nem kerül tárolásra. Ez teljesen szándékos, mivel nem tanácsos letölthető formában tárolni ezeket az adatokat, még akkor sem, ha az elérési út és a fájlnév "ismeretlen".



4.3.3. Beállítások

Ezen az oldalon adhatók meg a portál globális és az adminisztrációs felület beállításai. A beállítható tulajdonságok a modulok által bevezetett args.php féle megoldáshoz hasonlóan vannak rögzítve. Itt az $args, $tableSpecs és $options tömböket az args_portal.php és args_admin.php fájlokban található programkódok definiálják, amelyek a rendszer admin mappájában foglalnak helyet. Itt adhatók meg olyan JavaScriptek is, amelyek minden oldalon a window.onload esemény bekövetkezésekor, vagyis az oldal betöltődésekor elindulnak.

A beviteli űrlap a tulajdonságoknak, azok típusainak és specifikációinak megfelelően generálódik, így igazán könnyű a kitöltése. A beállítások mentése gombot megnyomva egy JavaScript kód legenerálja a módosítások érvénybe léptetéséhez szükséges CMD parancsot, majd elküldi a szervernek az oldal újratöltésével.

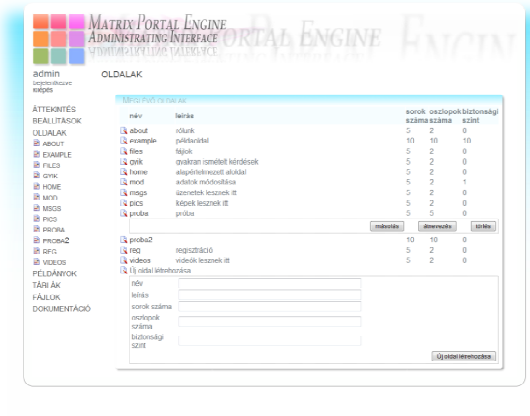


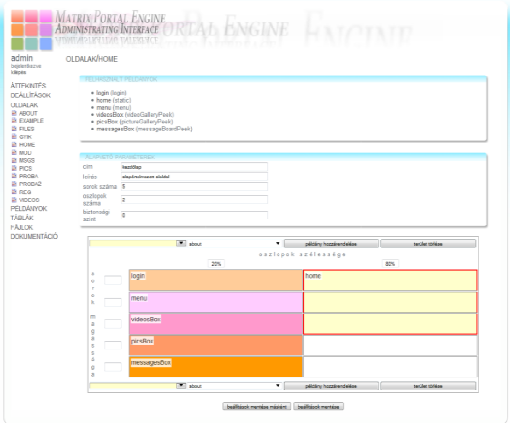
4.3.4. Oldalak

A felület oldalak című oldalán az oldalak kezelésére nyílik lehetőség. Ez az oldal egy fő- és több aloldalra tagolódik. Az aloldalak számát a rendszerben létrehozott oldalak száma határozza meg, minden oldalhoz tartozik egy aloldal. A főoldalon látható az összes oldal és alapvető fájlműveleteket lehet rajtuk végrehajtani, mint a törlés, másolás, átnevezés, valamint itt hozható létre új oldal is. A műveletek táblázat megfelelő sorának elején található szerkesztés gomb megnyomásával válnak láthatóvá.

Az aloldalakon az egyes oldalak, azok tulajdonságai szerkeszthetők. Az oldalon láthatók az oldalon található példányok, hogy azok melyik modulhoz tartoznak, az oldal alapvető tulajdonságait (cím, leírás, sorok és oszlopok száma és megtekintéséhez szükséges jogosultsági szint) tartalmazó űrlap és az oldalszerkesztő.

Az oldalszerkesztővel lehet az oldalt reprezentáló táblázat celláit egyesíteni, példányokat hozzárendelni és az egyes sorok és oszlopok szélességeit, magasságait beállítani. A példány hozzárendelés színek alapján történik, minden cellacsoport más színnel kell, hogy rendelkezzen, amely nem fehér. A színek ettől eltekintve szabadon választhatók a színválasztó legördülő listából. Egy cellacsoport létrehozásához válasszunk egy nem használt színt, egy fehér területen klikkeljünk a kívánt terület bal felső sarkába, majd a jobb alsóba. Ilyenkor a legördülő listában kiválasztott példány automatikusan hozzárendelődik létrehozott cellacsoporthoz, a módosításhoz a példány hozzárendelése használandó. A cellacsoportok nem fedhetik át egymást, így csakis fehér, azaz szabad területen hozhatunk létre cellacsoportot. Terület kijelöléséhez elég egy duplaklikk a területen, vagy az annak megfelelő szín kiválasztása. A törléshez a terület törlése gomb használható.





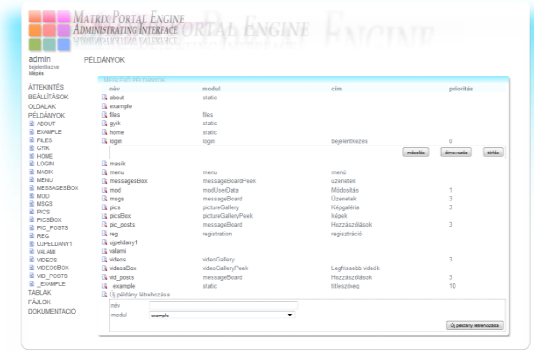
4.3.5. Példányok

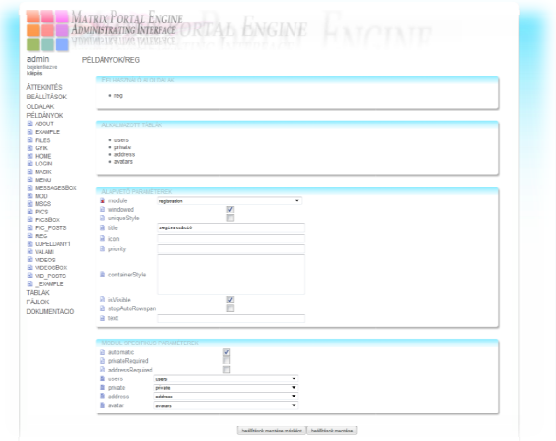
Az adminisztrációs felület példányok oldalán a példányok kezelésére nyílik lehetőség. Ez az oldal szintén egy fő- és több aloldalra tagolódik. Az aloldalak számát a rendszerben létrehozott példányok száma határozza meg, minden példányhoz tartozik egy aloldal. A főoldalon látható az összes példány és alapvető fájlműveleteket lehet rajtuk végrehajtani, mint a törlés, másolás, átnevezés, valamint itt hozható létre új példány is. A műveletek táblázat megfelelő sorának elején található szerkesztés gomb megnyomásával válnak láthatóvá.

Az aloldalakon az egyes példányok tulajdonságai szerkeszthetők. Láthatók a felhasználó aloldalak és az alkalmazott táblák és az űrlapok az alapvető és a modul specifikus tulajdonságok szerkesztéséhez. A tulajdonságok itt is típusosak, így az űrlapok kitöltése itt is segítve van, a beállítások oldal űrlapjaihoz hasonlóan.

A módosítások érvénybe léptetésére rendelkezésre áll egy mentés és egy mentés másként gomb is ugyanúgy, mint az "oldalak" aloldalai esetében is. A mentés másként gomb lehetőséget nyújt egy új beállítás kipróbálására anélkül, hogy a régi elveszne.

Az oldalon látható kapcsolódó elemek, vagyis az alkalmazott táblák és felhasználó oldalak természetesen az azokra mutató hiperhivatkozások, vagy népszerűbb nevén linkek. Ez nagy mértékben megkönnyíti a navigációt. Ezt a megoldást alkalmaztam a táblák és az oldalak aloldalai esetében is.





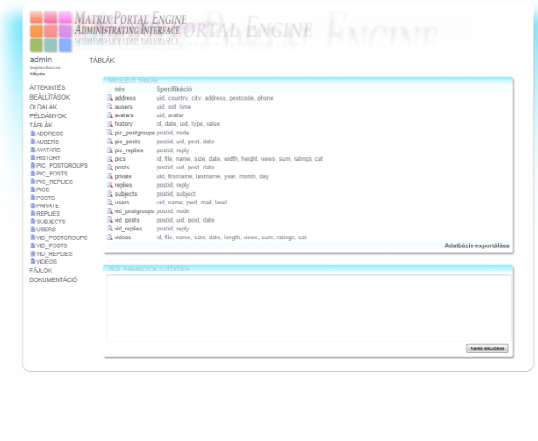
4.3.6. Táblák

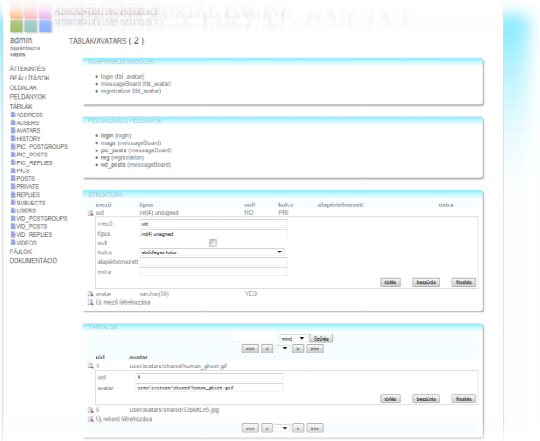
A táblák című oldalon az adatbázis táblák kezelésére nyílik lehetőség. Ez az oldal egy fő- és több aloldalra tagolódik. Az aloldalak számát a rendszerben létrehozott táblák száma határozza meg, minden táblához tartozik egy aloldal. A főoldalon látható az összes tábla és azok specifikációi, valamint egy SQL utasítások bevitelét lehetővé tevő szövegmező. A táblákon az egész táblát érintő műveleteket lehet végrehajtani, mint az üres tábla létrehozása, másolás, átnevezés, tartalom törlése és a törlés. A műveletek táblázat megfelelő sorának elején található szerkesztés gomb megnyomásával válnak láthatóvá. Új táblákat itt nem lehet külön létrehozni, mivel a rendszer rögzített szerkezetű táblákat kezel, amelyek hozzárendeléskor jönnek létre.

Az SQL beviteli mező segítségével lehet táblákat importálni, speciális lekérdezéseket végrehajtani és minden mást, amit lehetővé tesz.

Az aloldalakon az egyes táblák szerkezete és tartalma szerkeszthető. Láthatók a kompatibilis modulok és a felhasználó példányok, az utóbbiak természetesen hiperhivatkozásként.

Mind a tartalom, mind a struktúra táblázatosan van megjelenítve, amelynek megfelelő sorának első oszlopában lévő szerkesztés gombra klikkelve válik láthatóvá a szerkesztési terület, a módosító gombokkal. A módosító gombok lehetőséget adnak a szerkesztési területen található adatok rekordszintű beszúrására, módosítására és törlésére. Hasonló módon új rekord és mező is létrehozható. A tartalom mezők szerint kulcsszóval szűrhető és rendezhető, a keresések hatékonysága növelése érdekében.





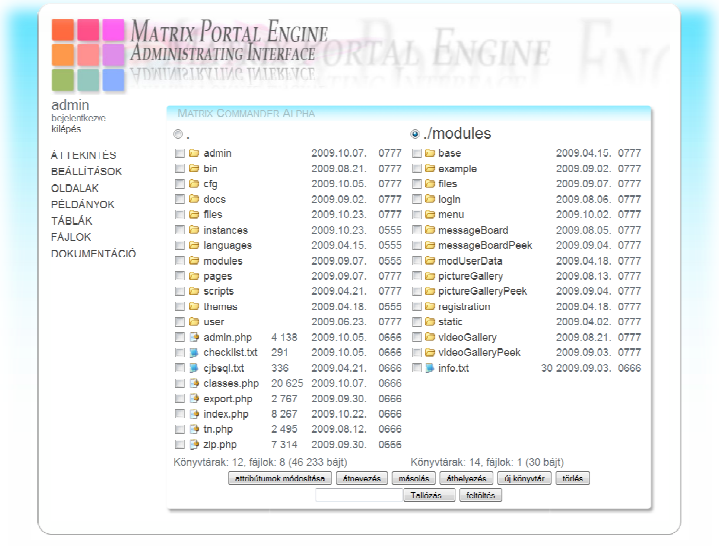
4.3.7. Fájlok

Szükségesnek találtam egy fájlok kezelésére alkalmas rész megírását is, ez kapott helyet a fájlok oldalon. Ez tulajdonképpen egy kétablakos X Commander-szerű fájlkezelő, amelyet ezen tulajdonsága nyomán Matrix Commander névre kereszteltem. A felület tradicionálisan két ablakra, azok címsorára és az azok alatt elhelyezkedő műveletek gombjaira tagolódik.

Az ablakok címsora tartalmazza az ablakokban listázott könyvtárak elérési útját hiperhivatkozásként, így biztosítva a visszafelé történő navigálást a fájlrendszer könyvtárfájában. Tartalmaznak még egy rádiógombot, amely az aktív ablakot jelöli meg.

Az ablakok táblázatosan tartalmazzák az egyes bejegyzések, vagyis könyvtárak és fájlok tulajdonságait (név, méret, módosítás dátuma, attribútumok) és az első oszlopban egy jelölődobozt, amely a kijelölésre szolgál. Könyvtárak nevére klikkelve bejuthatunk abba, fájlok nevére klikkelve azok megnyílnak egy új lapon vagy ablakban. Egy ablak összes elemének kijelölése, illetve ezek eltávolítása is lehetséges. Ez egy duplaklikkel eszközölhető a kívánt eredménynek megfelelő állapotú jelölőnégyzeten.

Az alkalmazható műveletek alapvetően lehetnek egy- és kétparaméteresek. Az előbbi típusú az aktív ablak kijelölt elemeivel, vagy azok elérési útjával operál, az utóbbi az aktív felől a passzív ablak felé hajtja végre feladatát. A jelen pillanatban elérhető egyparaméteres művelek az attribútumok módosítása, átnevezés, új könyvtár létrehozása, törlés és a fájlfeltöltés. A kétparaméteresek a fájlok másolása és áthelyezése műveletek.



4.3.8. Dokumentáció

A rendszer elvontsága, egyedi működése végett szükségesnek láttam a dokumentáció csatolását ehhez a felülethez. Ez tulajdonképpen nem is a felület egy oldala, hanem egy egyszerű HTML nyelven leírt dokumentum, amely egy másik lapon vagy ablakban nyílik meg. A megjelenítendő lap nyelve a rendszer nyelvbeállításától függ.

A dokumentáció tartalmazza a rendszer alapvető működésének, telepítésének, futtatásának és használatának leírását. Tartalma nagyjából hasonló, mint amit jelen szakdolgozatom keretein belül leírtam, persze sokkal egyszerűbb és rövidebb formában.

5. Összefoglalás

Úgy érzem sikerült elérnem a kitűzött célt, habár nem volt egyszerű, sokáig is tartott és még mindig akadnak apróbb hibák, hiányosságok a rendszerben.

A rendszerrel szemben támasztott elvárások első pontja az volt, hogy legyen moduláris felépítésű. Ez az elvárás szolgálja azt a célt, hogy a rendszer valóban széles körben legyen alkalmazható, legyen univerzális. A modulok egységes interfész segítségével kapcsolódnak a rendszerbe, amely semmiféle különbséget nem tesz köztük, csak futtatja őket a megadott adatok alapján. Elmondható, hogy a rendszer valóban moduláris, vagyis inkább teljesen modulorientált lett. A rendszer bármilyen feladatot el tud látni, ha rendelkezik a megfelelő modulokkal, vagy legalábbis felhasználási területe igen szélesre nyújtható.

Mivel egy CMS rendszerről van szó, elengedhetetlen tulajdonsága a tartalomkezelés biztosítása. Az adminisztrációs felületen, vagyis rendszer backendjén ez véghezvihető mindenféle szerkezeti módosításokkal együtt. Mivel a modulok funkcionalitása ismeretlen a rendszer számára, így a modul specifikus tartalomkezelési eszközök a frontenden helyezkednek el, de ez egyáltalán nem jelenthet problémát, hiszen pont ott van a rendeltetési helyük és a többszintű jogkörkezeléssel megakadályozható a jogosulatlan műveletek végrehajtása. Így lényegében ez a feladat is a modulokra hárul.

A frontend kinézete teljesen testreszabható lett, az adminisztrációs felület oldalszerkesztője segítségével bármilyen táblázatos elrendezés létrehozható, viszont a stílusmódosítások

alkalmazásához ismerni kell a CSS stílusleíró nyelvet, mivel stílusszerkesztő eszköz nem került a rendszerbe.

A rendszer tulajdonképpen az összes további célkitűzésnél felsorolt kritériumnak megfelel, a tulajdonságnevektől eltekintve többnyelvű, a támogatja a felszínválasztást, több egymástól megkülönböztetett jogkört támogat, futnia kéne szinte bármilyen PHP-MySQL konfiguráción és a lehető legtöbb böngésző képes megjeleníteni. Az adminisztrációs felületen könnyen lehet értékeket rendelni a típussal rendelkező tulajdonságokhoz, mivel az űrlapok annak megfelelően vannak generálva. Rendelkezik egy beépített, úgynevezett rich-text szövegszerkesztővel is, megkönnyítve az adminisztrációs feladatok végrehajtásának egy részét. Tulajdonképpen az eddigi tesztelések alapján úgy néz ki, hogy egy gyakorlatban is működőképes megoldásról van szó.

És most jöjjenek a negatívumok, a rendszer hiányosságai és hibái. Sajnos nem sok időm maradt a tesztelésre. Fejlesztés közben ugyan minden kódrészlet funkcionális tesztelését elvégeztem, de a teljes rendszert érintő funkcionális és nem funkcionális tesztelések igencsak részlegesek még.

Ami első körben megállapítható, hogy a rendszer nem lett igazán felhasználóbarát. Nem lehet vele két-három klikkeléssel portált építeni, de kezelése azért könnyen megtanulható. A tartalom kezelése a backend felületen megvalósítható, de szintén nem egyszerű feladat, mivel az adatbázistáblák közti kapcsolatokat konkrétan nem tudja kezelni. Így a használt adatbázis szerkezete egy nagy masszának tűnhet. Ennek kiküszöbölésére a modulok, melyek a táblák közti kapcsolatokat definiálják, definiálhatnának nézeteket is, amelyek alapján akár a szerkesztés is megvalósulhatna.

Egy másik probléma a kiválasztott nyelvek elavultsága. A HTML 4.01 már egy igencsak régi ajánlás, de azért a mai elvárásoknak megfelelő dokumentumok leírására még mindig megfelelőnek mondható. A PHP 4.1 viszont annyira nem is régi, de használatának a PHP 5-tel szemben számos hátránya van, mint például a fejletlen objektum kezelés és a hiányos objektum orientált eszközrendszer.

A modulok programkódján nincs megvalósítva semmiféle bezárás a rendszerhez képest, azzal azonos szinten futnak. Emiatt a modulfejlesztés egy kicsit bonyolultabb feladat, habár a

kódok függvényekbe zárhatók, még jobb lenne, ha a modulok osztályok és a példányok valódi példányok, azaz objektumok lennének. Az egyes modulkódok könnyen nyomon követhető módon építhetnének egymás kódjára és interfészek segítségével még rugalmasabb lehetne a rendszer. Eredetileg így született meg bennem ez a gondolat, de aztán letettem róla, mert úgy gondoltam, hogy így gyorsabb lesz a kód, mivel ezek kezelése azért csak komolyabb feladat. És ezzel sikerült egy újabb kényes kérdést érintenem, amely a sebesség. Összehasonlításra még nem kerítettem sort más rendszerekkel, de úgy érzem, hogy lehetne gyorsabb is. Tehát a kód optimalizálása még várat magára.

Kompatibilitási problémák is akadhatnak még a rendszerben. A program kódjait elméletileg a PHP 4.1 szintjén írtam, de csak PHP 5 alatt teszteltem, így előfordulhat, hogy néhány helyen javításra szorul. Továbbá nincs felkészítve eltérő futtatási direktívákkal konfigurált PHP értelmezőn való futtatásra sem, habár ez viszonylag egyszerű feladat, talán pont azért maradt ki.

Így a végére úgy gondolom, hogy az említett hibák és hiányosságok ellenére nem lett rossz a rendszer. Létre fogom tudni hozni és kezelni vele a saját személyes portálomat, amelyen eddigi és ezután következő munkáimat osztom majd meg a nagyközönséggel.

Ezúton szeretnék köszönetet nyilvánítani családomnak a sok türelemért és segítségért, témavezetőmnek a szakdolgozat elkészítésében nyújtott segítségéért és a Debreceni Egyetem tanárainak színvonalas előadásaikért.

6. Irodalomjegyzék

R. Allen Wyke, Jason Gillam, Charlton Ting: Pure JavaScript, Second Edition, SAMS, 2002

Matt Zandstra: PHP 24 óra alatt, Kiskapu Kft., 2001

Chelsea Valentine, Ed Tittel, Natanya Pitts: HTML 4, Kossuth Könyvkiadó, 2001

HTML 4 Reference, http://www.htmlhelp.com/reference/html40/

Guide to Cascading Style Sheets, http://www.htmlhelp.com/reference/css/

CSS Reference, http://www.w3schools.com/css/css_reference.asp

PHP manual, http://www.php.net/download-docs.php

MySQL Reference Manual, http://dev.mysql.com/doc/

JavaScript and HTML DOM Reference, http://www.w3schools.com/jsref/default.asp

JavaScript Kit - DOM Reference, http://www.javascriptkit.com/domref/

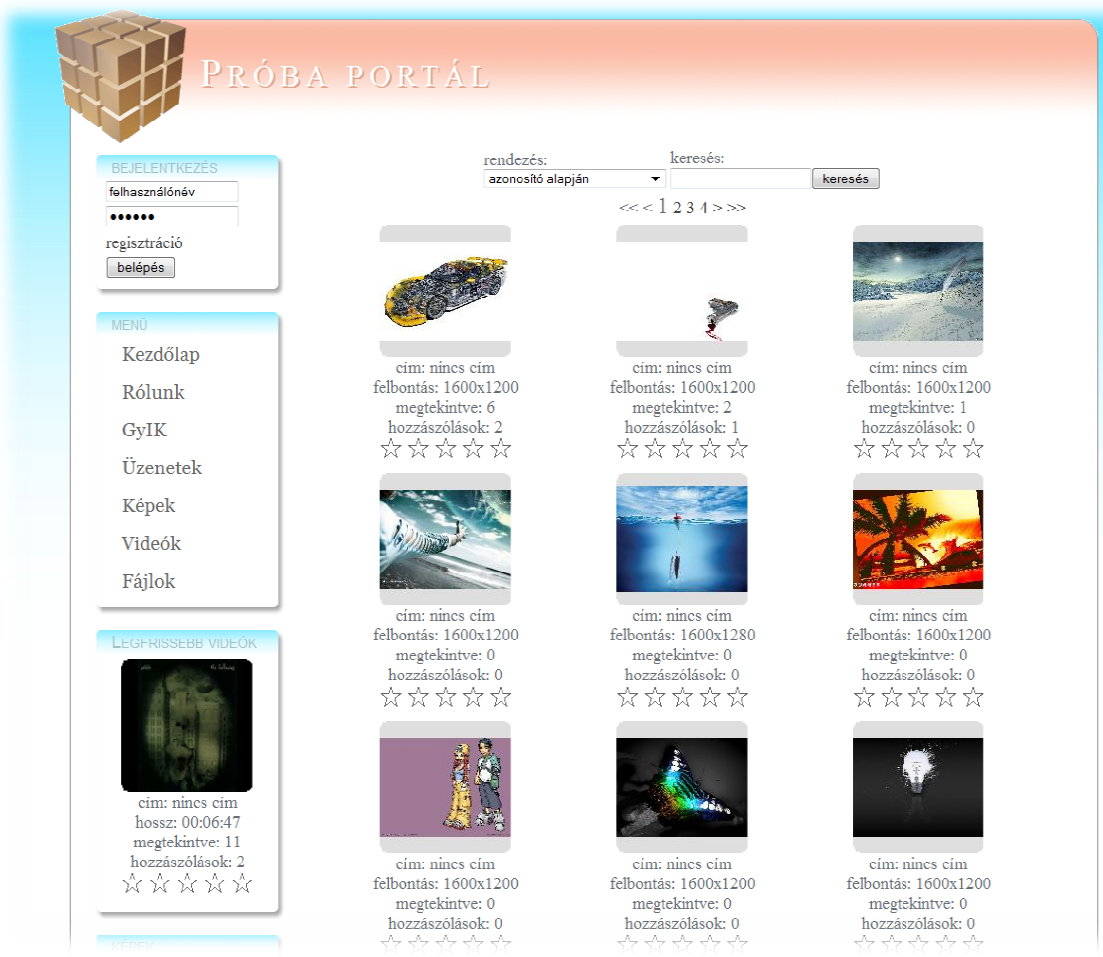
Nagy segítséget jelentettek továbbá a Wikipédia oldalai:

- http://en.wikipedia.org/wiki/Cascading_Style_Sheets
- http://en.wikipedia.org/wiki/Content_management_system
- http://en.wikipedia.org/wiki/HTML
- http://en.wikipedia.org/wiki/JavaScript
- http://en.wikipedia.org/wiki/MySQL
- http://en.wikipedia.org/wiki/SQL
- http://en.wikipedia.org/wiki/Client-side
- http://en.wikipedia.org/wiki/Document_Object_Model
- http://en.wikipedia.org/wiki/Usage_share_of_web_browsers
- http://en.wikipedia.org/wiki/Web_portal
- http://hu.wikipedia.org/wiki/Mátrix_(matematika)
- http://hu.wikipedia.org/wiki/PHP
- http://hu.wikipedia.org/wiki/SGML

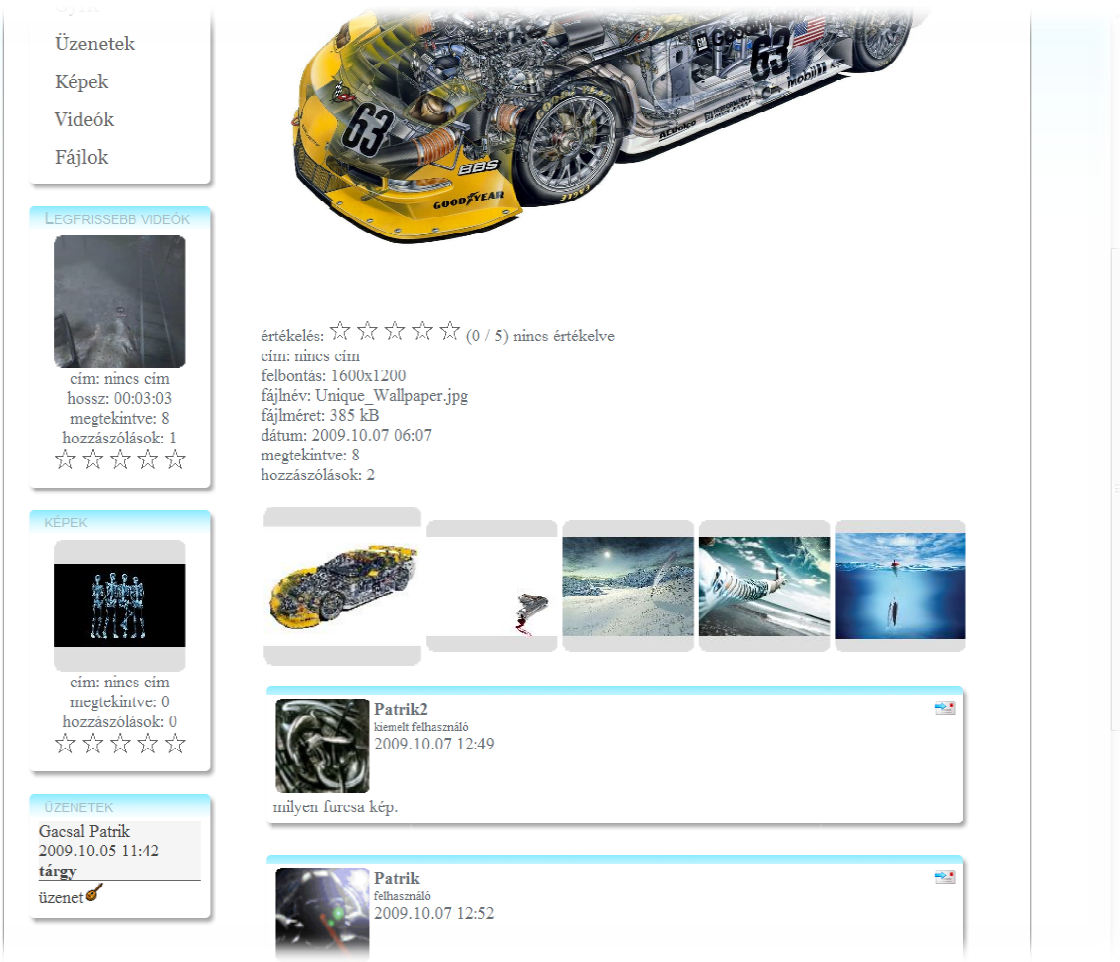
- http://hu.wikipedia.org/wiki/Tartalomkezelő_rendszer
- http://hu.wikipedia.org/wiki/World_Wide_Web_Consortium

7. Függelék

A rendszer képességeit bemutató "Próba Portál" egy oldala.



Egy másik oldal, itt a középpontban egy képgaléria modul kimenete látható.



Ezen az oldalon látható többek között egy képgaléria és egy üzenőfal modul kimenete, és azok együttműködése. A lent látható hozzászólások a fenti képhez tartoznak, a bal oldalon elhelyezkedő legfrissebb videók, képek és üzenetek blokkok a hozzájuk csatolt képgaléria, videógaléria és üzenőfal tartalmába nyújtanak betekintést.