Pécsi Tudományegyetem
Pollack Mihály Műszaki Kar
Műszaki Informatika Tanszék

SZAKDOLGOZAT

A WordPress tartalomkezelő rendszer bemutatása

Készítette: Mondovics Mihály
Konzulens: Bárdonicsek Róbert

Pécs
2007

PÉCSI TUDOMÁNYEGYETEM
POLLACK MIHÁLY MŰSZAKI KAR
Műszaki Informatika Tanszék
7624 Pécs, Rókus u. 2.

Szakdolgozat száma:

MI XVII.-169/2003/2007

SZAKDOLGOZAT FELADAT

Mondovics Mihály
hallgató részére

A záróvizsgát megelőzően szakdolgozatot kell benyújtania, amelynek témáját és feladatait az alábbiak szerint határozom meg:

Téma: A WordPress tartalomkezelő rendszer bemutatása

Feladat:

- a WordPress tartalomkezelő rendszer felépítése
- weboldalak programozása (HTML, PHP)
- sablon készítése a WordPress tartalomkezelőhöz
- beépülő plugin programozása a WordPress tartalomkezelőhöz

A szakdolgozat készítéséért felelős tanszék: PTE PMMK MIT

Témavezető: Bárdonicsek Róbert
Munkahelye: PTE PMMK MIT

Pécs, 2007. június 2.

dr.Szakonyi Lajos
tanszékvezető

HALLGATÓI NYILATKOZAT

Alulírott szigorló hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye. A felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Egyéb jelentősebb segítséget nem vettem igénybe.
Az elkészült szakdolgozatban talált eredményeket a főiskola, a feladatot kiíró intézmény saját céljaira térítés nélkül felhasználhatja.

Pécs, 2007. június 2.

.....

hallgató aláírása

Köszönetnyilvánítás

Köszönöm srácok ezt a négy évet. Nélkületek nem ment volna!

- Auer Zoltán
- Dénisch Péter
- Oszlár Gergely
- Patócs Zoltán
- Ragadics Éva
- Werner Norbert

TARTALOMJEGYZÉK

1. Bevezetés

Napjainkra az internet már szinte mindenki számára elérhetővé vált. A szolgáltatók közti nyílt verseny miatt a szélessávú, korlátlan előfizetések díja eléggé lecsökkent ahhoz, hogy már ne csak azok otthonában legyen internet, akiknek az létszükség, hanem azoknak is, akik csak megszeretnének ismerkedni az új dolgokkal. Ez a megnövekedett és nem éppen az informatika területén jártas felhasználói tábor követelte meg a weben az olyan szolgáltatások megjelenését, amik eléggé egyszerűek és felhasználóbarátak ahhoz, hogy bárki komolyabb szaktudás nélkül tudja használni.
Az internetező ember egy idő után már nem éri be az interneten található tartalmak „fogyasztásával”, Ő maga is szerzővé akar válni. Ezt a folyamatot könnyítik meg a bárki számára ingyenesen elérhető nyílt forráskódú tartalomkezelő rendszerek.
A WordPress alapjában véve egy blogokhoz készített tartalomkezelő, de a számos elérhető plugin és kiegészítő segítségével egy komolyabb feladatok ellátására alkalmas információs rendszert is kihozhatunk belőle.
A szoftver használata rendkívül egyszerű, de a rendszer megértése és fejlesztése megkövetel némi szaktudást. Ezen ismeretek elsajátításához kívánok némi segítséget nyújtani a dolgozatommal.

2. A tartalomkezelőkről általában

A tartalomkezelő rendszerek (angolul Content Management System, CMS) olyan szoftverrendszerek, amelyek a nem strukturált információkat (mint például az internetes portálok), akár több felhasználó általi elkészítését, kezelését, és tárolását segítik. Tartalomnak (content) nevezzük azon információtöredékek halmazát, amelyet összerakva összefüggő, felismerhető "dolgot" kapunk. Ilyen például egy újságcikk, mely a címből, rövid összefoglalóból, a cikk törzséből és szerzőjéből állhat. Emellett tartalmazhat például képeket, referenciákat, kitérőket, hivatkozásokat, amelyek hozzáadnak valamit, gazdagítják azt, több értéket nyújtva az olvasónak.

A tartalomkezelő rendszerek célja az ügyfél online megjelenései egyszerű kezelésének lehetővé tétele. A tartalomkezelő rendszerek jellemzője az egyszerű, laikusok által is könnyen elsajátítható működtetés. A különféle CMS rendszerek képesek átfogni egy vállalat internet/intranet-es megjelenéseit, és egységesen kezelni azokat.
A CMS-ek legfontosabb tulajdonsága a dinamizmus. A tartalmat, annak megjelenését megfelelő jogosultság esetén bármikor, bárhonnan meg lehet változtatni.
A tartalomkezelő rendszer működésének lényege az alábbiakban foglalható össze: a rendszer felhasználói (adatgazdák) a változtatni vagy bővíteni kívánt tartalmi egység adminisztrációs felületén keresztül feltöltik az adatbázisba a frissítendő tartalmi elemeket. A rendszer a feltöltött tartalmakat – szükség esetén jóváhagyatás után – az előre meghatározott arculati sablonba beépítve megjeleníti a honlapon. Amennyiben új oldal került feltöltésre, úgy az oldal linkje automatikusan megjelenik az oldaltérképen és a menükben is.

A rendszer egy böngészőablakon keresztül elérhető, ezért nincs szükség kliensoldali szoftverek telepítésére, ezek használatának megtanulására, csupán a webes böngésző alapszintű ismerete szükséges.

3. PHP

A PHP elterjedt nyílt forráskódú szerveroldali programozási nyelv. Szintaktikája leginkább a C programozási nyelvéhez hasonlít. Megalkotója Rasmus Lerdorf. Mára azonban egy egész csapat foglalkozik a nyelvvel.
Saját magára utaló (rekurzív) mozaikszó, az angol: *PHP Hypertext Preprocessor* kifejezésből ered. (A név eredeti jelentése az angol *Personal Home Page*, azaz a személyes honlap kifejezés rövidítése volt.)
A **Zend Technologies** a PHP mögött álló első számú cég, a PHP alapjainak készítői indították.
Az értelmezőt támogató kereskedelmi termékeket fejlesztenek, terméktámogatással. A honlapjukon található referenciák egyértelművé teszik a PHP egyre szélesedő

elfogadottságát a magasabb üzleti szektorban. A PHP továbbra is ingyenes termék, minden feladatra elérhetőek költségmentes megoldások.
A PHP oldalak elkészítésénél az eredmények formázására a HTML-t használják. Amikor egy PHP-ben 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 segítségével olyan összetett alkalmazásokat is készíthetünk, amelyre az ügyféloldali szkriptek nem képesek (vagy ha igen, korlátozottan). Ilyen pl. a bejelentkezés, az adatbáziskezelés, fájlkezelés, kódolás, adategyeztetés, kapcsolatok létrehozása, e-mail küldése, adatfeldolgozás, dinamikus listakészítés stb. Minden olyan esetben, ahol nagyszámú ismétlődő feladatsort kell végrehajtani (pl. képek listázása és linkelése, listakészítés stb.), ott ez a programnyelv nagyszerű segítség.
A PHP programok futhatnak közönséges (parancssori) programként is, nem HTML oldalba építve. Ezt azonban ritkán használják.

4. WordPress

4.1. Mi is az a WordPress?

A WordPress egy elegáns, jó felépítésű személyes publikálási rendszer PHP és MySQL alapokon építve, a GPL licenc alatt kiadva. A b2/cafelog hivatalos utódja. A WordPress egy viszonylag új portálmotor, de gyökerei és fejlesztése egészen 2001-ig visszanyúlik.
A WordPress tartalomkezelő helyes működéséhez legalább 4.2-es verziószámú PHP szerverre és minimum 3.23-as verziójú MySQL szerverre van szükség.

4.2. A WordPress felépítése

4.2.1. Fájlok

A WordPress tartalomkezelő rendszer fájljai egy jól strukturált és egyszerű könyvtárszerkezetben találhatóak. Itt funkcióik szerint vannak különválasztva a fájlok. A struktúra a következő:

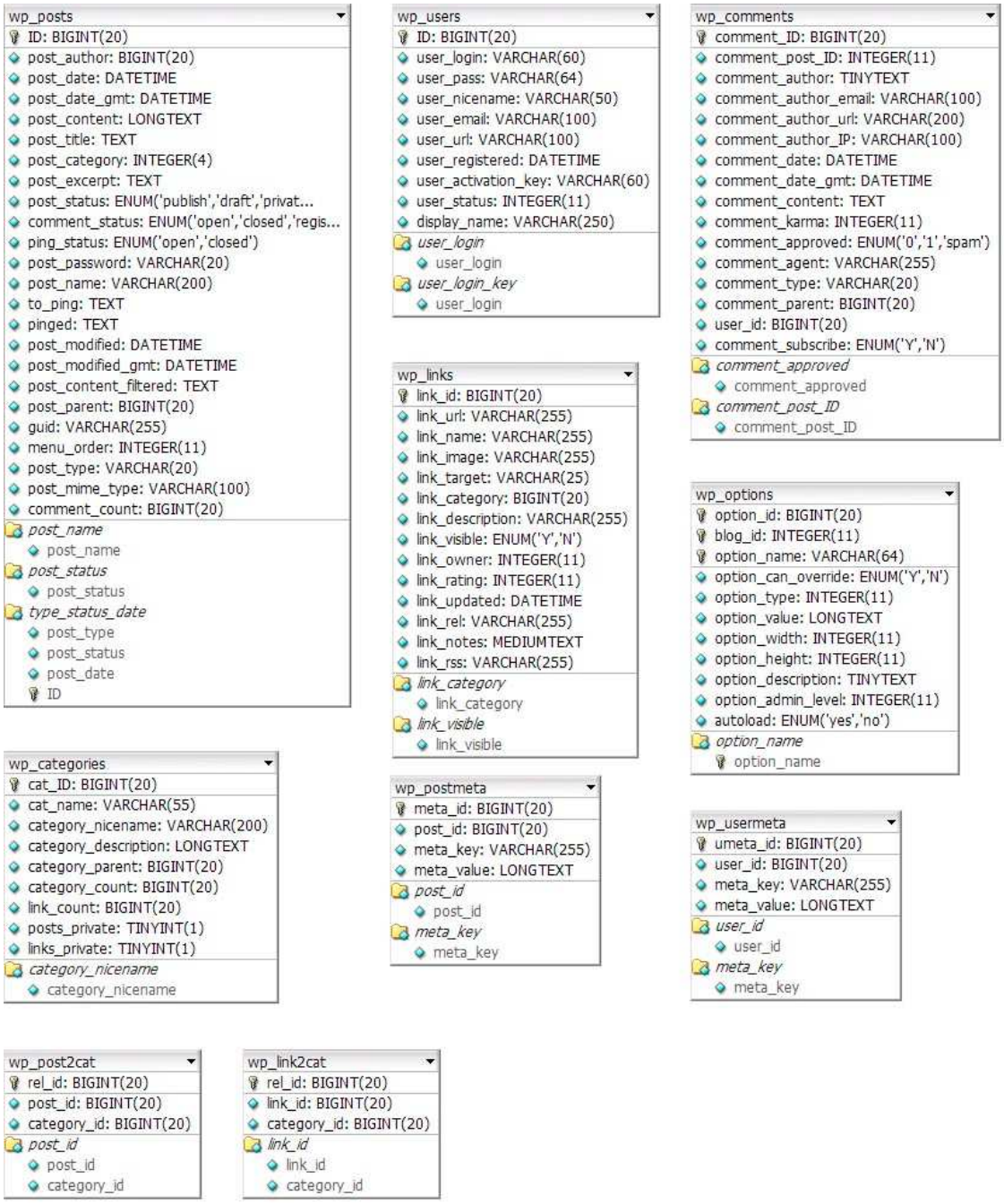
- wp-admin
- wp-content
    - plugins
    - themes
    - uploads
- wp-includes

Wp-admin:
Ebben a mappában az adminisztrációhoz szükséges rendszerfájlok és függvényleírások találhatóak.

Wp-content:
A felhasználó mappája. Itt találhatóak a pluginek, sablonok és a feltöltött állományok is. A helyes működés érdekében erre a mappára írási jogot kell adnunk.

Wp-includes:
Az oldal működéséhez szükséges függvények definícióit tartalmazó fájlok lelőhelye.

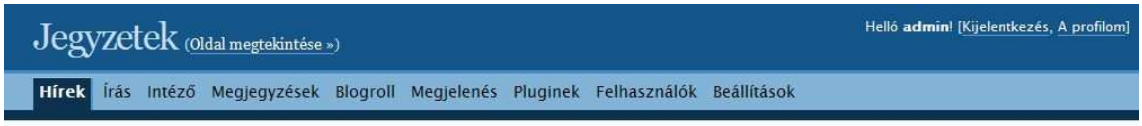
4.2.2. Adatbázis



1. ábra: A WordPress adatbázisa

4.2.3. Admin

Minden tartalomkezelő alapjában véve felbontható két fő részegységre. Az úgynevezett *frontend* a nyilvánosságnak szól. Ezzel a felülettel találkozik minden felhasználó, aki ellátogat az adott oldalra. Ez tulajdonképpen maga az oldal. A másik főegység a *backened*, amit csak kiemelt jogú felhasználók érhetnek el. Itt történik az oldalon megjelenő tartalom menedzselése és karbantartása, valamint az adminisztrátori feladatokat is itt lehet elvégezni.
A felhasználói jogok határozzák meg, hogy ki milyen szinten avatkozhat bele az oldal működésébe.
A WordPress rendszer admin felülete az alábbi logikai egységekre és alegységekre van bontva (az alacsonyabb jogokkal rendelkező felhasználók elől némely menüpontok rejtve vannak):



2. ábra: Az admin oldal menüpontjai

- **Hírek**: amikor belépünk az admin oldalra, akkor alapesetben ide irányít át minket a rendszer. Itt gyors összefoglalót látunk a legfrissebb hozzászólásokról és bejegyzésekről, valamint a WordPress szerverről ide töltődnek le a legfrissebb fejlesztő információk.
- **Írás**
  - **Bejegyzés**: publikációk megírásáért és módosításáért felelős rész.
  - **Oldal**: aloldalak megírásáért és módosításáért felelős rész.
- **Intéző**
  - **Bejegyzések**: az adatbázisban lévő publikációk táblázatos listája.
  - **Oldalak**: az adatbázisban lévő aloldalak táblázatos listája.
  - **Feltöltések**: feltöltött állományok menedszeléséért felelős modul.
  - **Kategóriák**: új kategóriákat hozhatunk itt létre, valamint a már meglévőket is itt módosíthatjuk.

  - **Fájlok**: a szerveren lévő fájlokat nyithatjuk meg itt szerkesztésre.
  - **Import**: bejegyzések és beállítások importálása.
  - **Export**: bejegyzések és beállítások exportálása.
- **Megjegyzések**
  - **Megjegyzések**: összes megjegyzés listázása.
  - **Moderációra várakozók**: moderációra várakozó megjegyzések kezelése.
- **Blogroll**
  - **Blogroll**: felvett linkek táblázatos formája.
  - **Új link hozzáadása**: új link felvétele.
  - **Linkek importálása**: linkek importálása külső adatbázisból.
- **Megjelenés**
  - **Sablonok**: az oldal számára elérhető sablonok listája. Itt választhatjuk ki az éppen használni kívánt felületet.
  - **Sablonszerkesztő**: az elérhető sablonok fájljainak módosítását teszi lehetővé.
- **Pluginek**
  - **Pluginek**: az oldal számára elérhető pluginek listája. Itt kapcsolhatjuk ki és be a kívánt kiegészítőket.
  - **Pluginszerkesztő**: az elérhető pluginek forráskódjának módosítására szolgáló felület.
- **Felhasználók**
  - **Felhasználók**: regisztrált felhasználók listája. Adminisztrátori szerepkörök csoportos módosítására itt van mód.
  - **Személyes profil**: regisztrációnkhoz tartozó személyes információinkat (név, jelszó, email, stb…) tarthatjuk itt karban.
- **Beállítások**
  - **Általános**: általános beállítások mint például: oldal neve, url-je, időzóna, stb…
  - **Írás**: néhány alapbeállítás a publikációk létrehozásával kapcsoltban

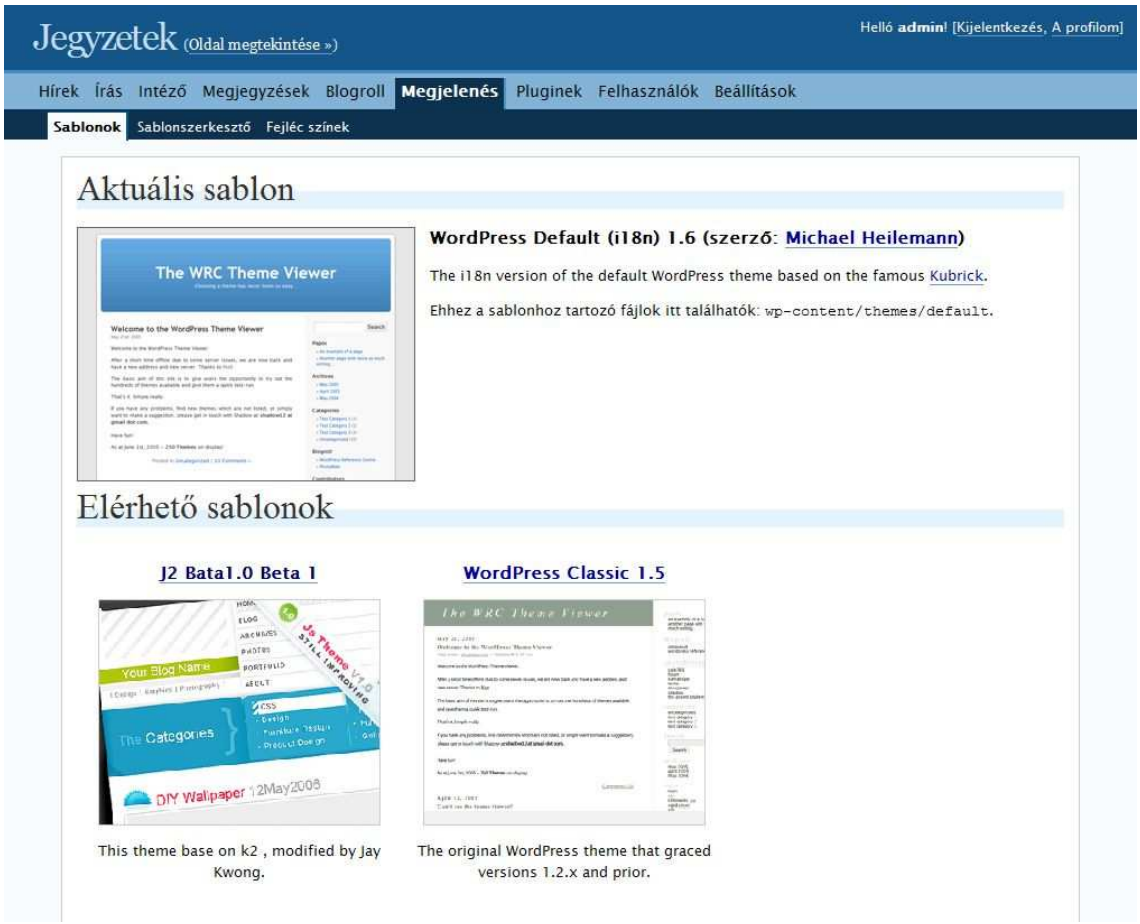
- **Olvasás**: itt beállíthatjuk a főoldalt, illetve azt, hogy hány publikáció jelenjen meg egyszerre. Módunk van az RSS kimenet finomhangolására is.
- **Interakciók**: bejegyzésekhez kapcsolódó interakciók (hozzászólás engedélyezés, pingback engedélyezés, stb) beállítása. Ezeket az opciókat természetesen a bejegyzés írásakor is megtudjuk határozni külön az aktuális publikációnkhoz is. A megjegyzések automatikus moderációjának beállítására is itt van lehetőségünk.
- **Keresőrobotok**: a különböző keresőrobotok blokkolásának beállítása
- **Permalinkek**: az állandó linkek formátumát határozhatjuk meg. Figyelem! Ha a szerverünkön nincsen engedélyezve a *.htaccess rewrite szabály*, akkor az oldalunk nem lesz elérhető ha bekapcsoljuk a permakinkeket.
- **Egyéb**: többek között a feltöltési könyvtár helyét adhatjuk meg itt.

4.3. Sablonkezelés

A tartalomkezelők tervezésének egyik legfontosabb tulajdonsága, hogy az adattartalom, a forráskód valamint a megjelenésért felelős részek jól elkülöníthetőek legyenek egymástól, és bármelyik cseréje ne befolyásolja a másik kettő működését.
A WordPress-ben ez a hármas tagolás jól megfigyelhető. Az adatok teljes mértékben az adatbázisban tárolódnak. A működést befolyásoló kódok is külön könyvtárban tárolódnak. Ezen forráskódoknak nincsen HTML kódokkal formázott kimenetük. A megjelenésért teljes mértékben a *themes* mappa fájljai a felelősek.
Az interneten rengeteg sablon érhető el teljesen ingyen a WordPress-hez. Ezek beüzemelése rendkívül egyszerű:

1. töltsük le a sablonfájlokat tartalmazó mappát
2. ezt a mappát másoljuk be a *wp-content/themes* mappába
3. jelentkezzünk be adminisztrátorként a WordPress-be és válasszuk ki a *megjelenés* menüpontot

4. itt látjuk az aktuális sablont, valamint a többi *wp-content/themes* mappában található sablonokat is. Azt is amit az imént másoltunk fel.
5. kattintsunk a kívánt sablon nevére.



3. ábra: Sablonválasztó

Ha most megnézzük oldalunk felületét, akkor láthatjuk, hogy immár a kiválasztott sablon szerint rendeződik a tartalom.

Sablonokat bárki készíthet a rendszerhez, ehhez csupán némi HTML és CSS tudás szükséges. Az alaplépéseket az 5.4-es pontban fogom tárgyalni.

4.4. Pluginkezelés

Az alap WordPress-be a fejlesztők csak a legszükségesebb CMS funkciókat tették bele. Az interneten viszont több ezer plugin található, így számos extra funkcióval bővíthetjük rendszerünket. A pluginek beüzemelése sokban hasonlít a sablonok munkára bírására.

1. töltsük le a pluginfájlokat tartalmazó mappát
2. ezt a mappát másoljuk be a *wp-content/plugins* mappába
3. jelentkezzünk be adminisztrátorként a WordPressbe és válasszuk ki az *pluginek* menüpontot
4. itt látjuk az összes elérhető plugin nevét és leírását, valamint azt, hogy éppen aktív-e vagy sem. A plugineket itt tudjuk ki-be kapcsolgatni. Aktiváljuk hát a most felmásolt pluginünket.

4. ábra: Elérhető pluginek be- illetve kikapcsolása

Az egyszerűbb kiegészítők ekkor rögtön működnek, de vannak olyanok is, amiknek némi finomhangolásra van szükségük. Ezek a pluginek általában létrehoznak egy saját menüpontot az *admin főmenük* közt, vagy a *beállítások* almenüi közt. Ezt a plugin leírása mindenféleképpen tartalmazza.

Kiváló pluginlelőhely például a WordPress központi Plugin Directory-ja:

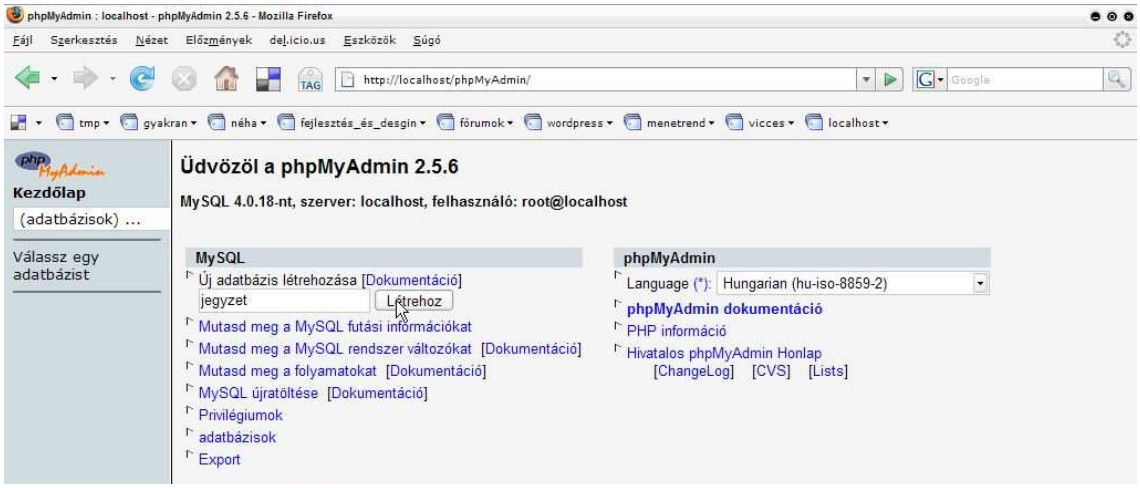
http://wordpress.org/extend/plugins/

5. A hallgatói információs rendszer létrehozása

Egy olyan rendszert fogunk létrehozni a WordPress segítségével, amely a hallgatókkal való kapcsolattartást, valamint a jegyzetek megosztását könnyíti meg. A különböző tárgyaknak megfelelően vesszük fel a kategóriákat, és ezekbe publikáljuk bejegyzések formájában a jegyzeteket, valamint a fontosabb információkat. Így a főoldalon vegyesen időrendben látszanak a bejegyzések, ám akit csak a saját szakának információi érdeklenek, azok egyetlen kattintással tudják szűrni az oldalon lévő információkat. A különböző publikációk körül párbeszéd alakulhat ki a hallgatók és oktatóik közt, így egy jól strukturált tudásbázis is létrehozható.
A feladat elvégzésére egy 2.1-es verziójú WordPress-t fogok használni, amiben már alapból benne vannak a nyelvi fájlok. Mindenesetre a későbbiekben tárgyalom azt is, hogy hogyan lehet nyelvileg lokalizálni a rendszerünket.

5.1. A WordPress telepítése

A WordPress telepítése rendkívül egyszerű. Nem igényel programozási ismereteket, hiszen majdnem minden automatizálva van benne. Legelső lépésként töltsük le a legfrissebb csomagot a http://wordpress.org/downloads helyről. Ha azt szeretnénk, hogy már a telepítő is magyar nyelvű legyen, akkor látogassunk el a http://word-press.hu oldalra, és a letöltések rovaton belül keressük ki a legfrissebb csomagot, amelyet elláttak a magyar nyelvi fájlokkal is. Az említett weboldalakon egy tömörített fájlt fogunk találni. Ezt kell kicsomagolnunk, majd a teljes tartalmat felmásolni egy tárhelyre. Erre a célra célszerű valamilyen FTP kezelő programot használni.
Második lépésként létre kell hoznunk az adatbázist, amiben majd a telepítő létrehozza nekünk automatikusan a táblákat amikben később az adatainkat fogjuk tárolni. Erre a feladatra célszerű a PHP MyAdmin-t használni.



5. ábra: A „jegyzet” nevű adatbázis létrehozása a szerveren

Ezután meg kell adnunk az adatbázis kapcsolódási adatait. Ehhez nyissuk meg az immáron a tárhelyünkön lévő *wp-config-sample.php* fájlt egy egyszerű szerkesztővel. Keressük meg a fájlban az alábbi kódrészletet és módosítsuk a tartalmukat:

```
// ** MySQL settings ** //
define('DB_NAME', '');    // The name of the database
define('DB_USER', '');     // Your MySQL username
define('DB_PASSWORD', ''); // ...and password
define('DB_HOST', '');    // 99% chance you won't need to change
this value

// You can have multiple installations in one database if you
give each a unique prefix
$table_prefix  = 'wp_';   // Only numbers, letters, and
underscores please!
```

Adatainkat az üres aposztrófok (’’) közé írjuk

- DB_NAME: az előző lépésben létrehozott adatbázis nevét kell megadnunk
- DB_USER: a MySQL szerverhez tartozó felhasználónevünket kell itt megadni
- DB_PASSWORD: a MySQL szerverhez tartozó jelszavunkat kell itt megadni
- DB_HOST: a MySQL szerverünk címét kell megadnunk

Módosíthatjuk a táblák előtagját is a `$table_prefix` változó értékének megváltoztatásával. Erre például akkor lehet szükségünk, ha ugyan abba az adatbázisba egyszerre kettő vagy több WordPress-t telepítünk.
(Ingyenes tárhelyszolgáltatók egy felhasználónak csak egy adatbázis létrehozását engedélyezik).
Figyelem! A rendszer különbséget tesz a kis- és nagybetűk között.
Amennyiben módosítottuk a fájl tartalmát, akkor mentsük el *wp-config.php* néven. A *wp-config-sample.php* fájlt letörölhetjük a szerverünkről.
Következő lépésként futtatnunk kell a WordPress telepítő fájlt. Ez az *install.php* lesz. A mi esetünkben ez a http://localhost/jegyzet/wp-admin/install.php cím. Ekkor egy rövid tájékoztatást fogunk látni a böngészőben. Kattintsunk az „*1. lépés*” linkre. A következő oldalon meg kell adnunk az oldalunk nevét, valamint az e-mail címünket. Most kattintsunk a „*Tovább (2. lépés)*” gombra. Ekkor a háttérben létrejönnek az adatbázisunkban a WordPress működéséhez szükséges táblák, valamint létrejön az admin felhasználó is. Ez a felhasználó teljes jogosultsággal rendelkezik. A következő képernyőn generálódik neki egy véletlen jelszó. Jegyezzük meg ezt a jelszót, majd a „*bejelentkezés*” linkre kattintva (vagy a http://localhost/jegyzet/wp-login.php címen) jelentkezzünk be az *admin* felhasználónévvel és a hozzá kapott jelszóval. Ezt a véletlen karakterekből álló jelszót célszerű rögtön megváltoztatni („*Felhasználók*” menüpont) valami olyanra, amit nem fogunk elfelejteni.
Oldalunk ezzel elkészült.

5.2. Honosítás

A nyelvi lokalizáció nagyon fontos, hiszen ezzel az idegen nyelven nem értő felhasználóink dolgát könnyítjük meg. Amennyiben a WordPress csomagunk nem tartalmazza alapesetben a nyelvi fájlokat, lehetőségünk van külön is feltölteni azokat.
A magyar nyelvi fájlokat a http://word-press.hu oldalon találhatjuk meg. A nyelvi fájlt (*.mo) be kell másolnunk a *wp-includes/languages* mappába. Ezután tudatnunk kell a WordPress-szel, hogy melyik fájlt használja. Ezt a *wp-config.php* fájl kézi

módosításával kell megtennünk. Nyissuk meg a fájlt egy szerkesztővel, és keressük meg az alább sort:

```
define ('WPLANG', '');
```

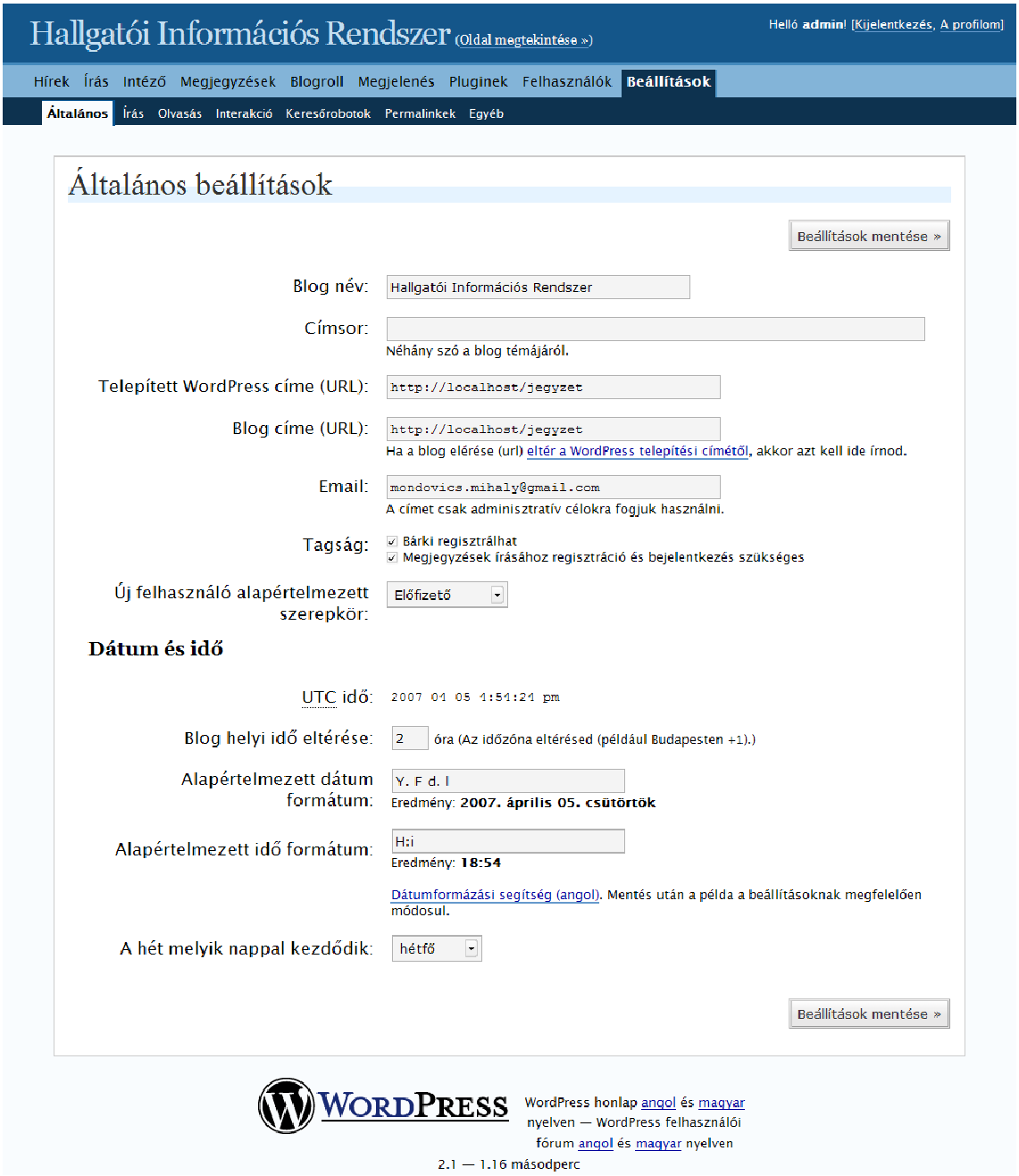
Az üres aposztrófok közt adjuk meg a nyelvi fájl nevét kiterjesztés nélkül. (pl.: ha a fájl a *hu_HU_2.mo* nevet viseli, akkor az aposztrófok közé a *hu_HU_2* karaktersort írjuk)
Ez a nyelv egy szótárfájl, ebből keresi ki a megfelelő sztringekhez tartozó párt a WordPress. Ezt függvényhívások segítségével végzi el, de ha olyan sablont használunk, amelynek a készítője nem függvényhívásokkal oldotta meg a szöveget, hanem beírta kézzel, akkor ezek a részek az eredeti nyelven fognak megjelenni.

5.3. Konfigurálás

Az oldal telepítése elkészült, ám még el kell végezni néhány beállítást. Ebben a részben alakítom ki az oldal szükséges funkcióit, hogy elláthassa feladatát. Az oldalt arra fogjuk használni, hogy megoszthassuk a hallgatókkal a különböző tantárgyakhoz tartozó jegyzeteket, valamint hogy hatékonyan tudjuk velük tartani a kapcsolatot. Ennek érdekében néhány extra plugint fogok munkára bírni, valamint készítek egy sajátot is. A sablon forráskódjába is betekintek, hogy egyedivé szabjam az oldalak működését.

5.3.1. Beállítások

Felhasználónévvel és jelszóval jelentkezhetünk be (http://localhost/jegyzet/wp-admin), majd végezzük el az alapbeállításokat. A *Beállítások* menüpont alatt találhatóak a finomhangoláshoz szükséges aloldalak:



6. ábra: Általános beállítások

Néhány szó a fontosabb opciókról:

Blog Név

Az itt megadott név fog látszani az oldal fejlécében és a böngészőablak címsorában is.

E-Mail

Erre a címre fogja küldeni a rendszer a jelentéseket új felhasználók regisztrálásáról, vagy a hozzászólásokról.

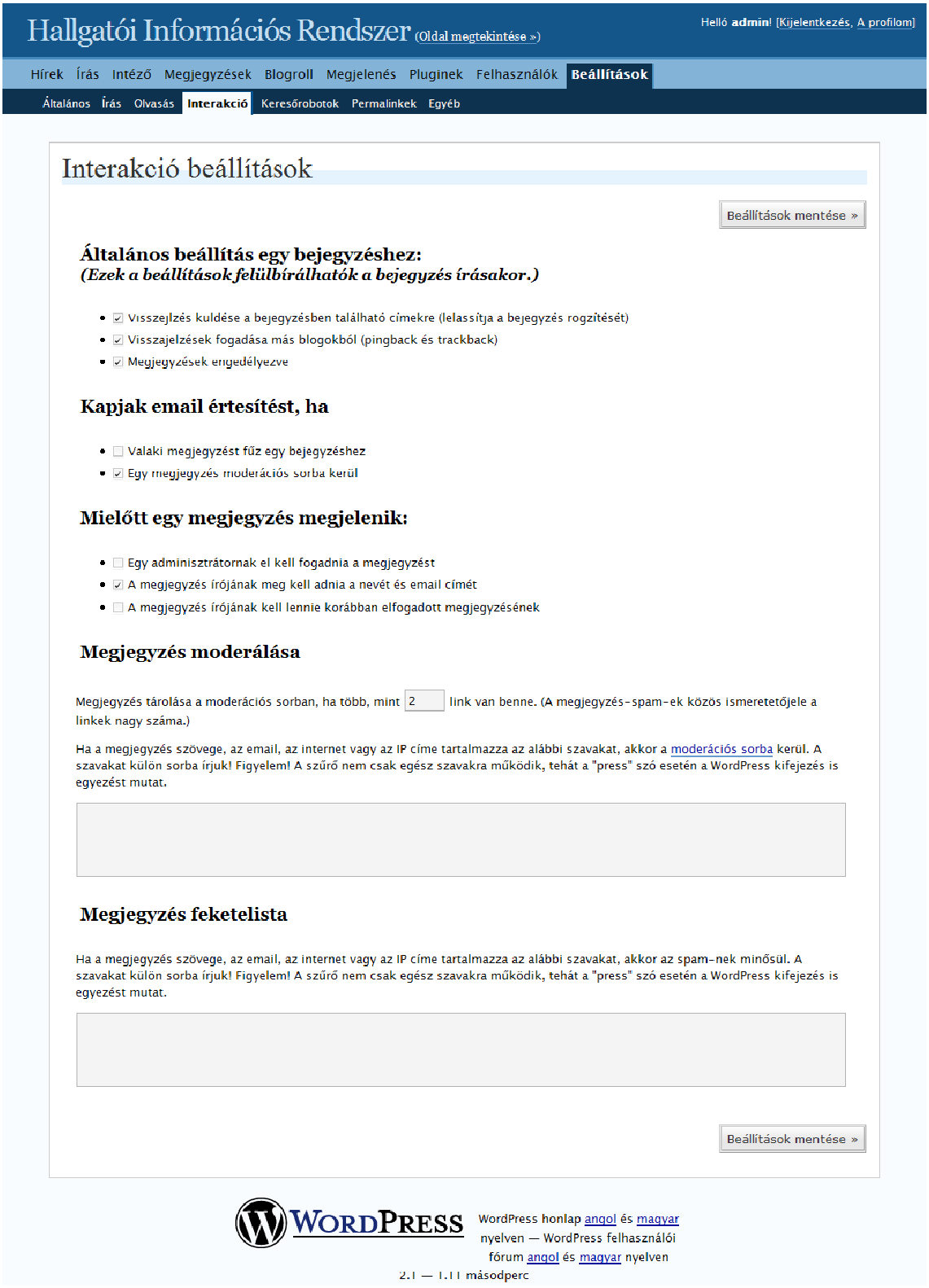
Tagság

Itt jelöljük be mindkét jelölőnégyzetet, mert így bárki regisztrálhat a rendszerbe, illetve megjegyzéseket csak a regisztrált felhasználók fognak tudni készíteni. Ez utóbbinak két fontos jelentősége van. Az egyik, hogy így némileg ellenőrizhetőek és szűrhetőek (bannolhatóak) lesznek a látogatók. Ez azért jó, mert egy hivatalos oldalt szeretnénk üzemeltetni, és nem volna jó, ha mindenféle dolog megjelenne és bárki teleszórhatná nem odaillő megjegyzéseivel az oldalunkat. Másik előnye a dolognak, hogy a spambotok (algoritmusok, amelyek autómatikusan generálnak reklámokat tartalmazó megjegyzéseket) nem fognak tudni megjegyzést létrehozni.

Új felhasználó szerepkör

A WordPress-ben 5 felhasználói szint létezik:

- **Előfizető**: semmilyen formában nem tudja módosítani az oldalunk tartalmát, kizárólag kommentálni tud.
- **Közreműködő**: tud bejegyzéseket írni, de nem publikálhatja azokat, kizárólag piszkozatként mentheti el. Egy magasabb jogú felhasználónak kell engedélyezni a megjelenést.
- **Szerző**: tud bejegyzést publikálni és a sajátját szerkeszteni.
- **Szerkesztő**: bejegyzéseit tudja publikálni, valamint a mások által írt publikációkat tudja szerkeszteni. A kategóriákat, linkeket is és mások megjegyzéseit is képes szerkeszteni, törölni. Gyakorlatilag csak az oldal paramétereit nem tudja módosítani.
- **Adminisztrátor**: az rendszer bármely paraméterét tudja módosítani.



7. ábra: Interakció beállítások

A többi bejegyzést hagyhatjuk az alapértelmezett értéken.

5.3.2. Kategóriák létrehozása

Minden publikációt kategóriákba fogunk szervezni. A főkategóriák lesznek a szakok nevei, az alkategóriák pedig a szakokhoz tartozó tantárgyak lesznek. Ezzel egy jól átlátható struktúrát kapunk.
A 2.1-es verziótól kezdve a fejlesztők összevonták a bejegyzések kategóriáit és a linkek kategóriáit. Így most már csak egyszerűen kategóriák vannak, amikbe egyaránt tudunk bejegyzést publikálni, és linkeket is felvenni. Egyáltalán nem értem mi ennek a szemléletnek az előnye, így kevésbé átláthatóbb szerintem. Valószínűleg arra gondoltak, hogy a kategóriákat a manapság olyan divatos címkéknek kell elképzelni, és így tudjuk a tartalmakat nem kategorizálni, hanem címkézni. Viszont a címkézés nem egyenértékű a kategorizálással, márpedig ezen oldal tartalmának áttekinthető tárolására és megosztására most kategóriák kellenek.
A probléma megoldása érdekében a sablont kell majd kicsit módosítani, hogy ott is ésszerűen jelenjenek meg az adatok. Erre a sablonkészítésnél kitérek.
Az alábbi logikai kategóriastruktúrát alakítsuk ki:

- Hírek
- Jegyzetek
    - Szak 1 (főkategória)
        - Szak 1-hez tartozó tantárgy 1 (alkategória)
        - Szak 1-hez tartozó tantárgy 2 (alkategória)
        - Szak 1-hez tartozó tantárgy *n* (alkategória)
        - …
    - Szak 2 (főkategória)
        - Szak 2-höz tartozó tantárgy 1 (alkategória)
        - …
    - Szak *n* (főkategória)

Lehetséges, hogy egy kurzus ugyanúgy előfordulhat két különböző szaknál is azzal a különbséggel hogy a tananyag más az adott szaknak megfelelően. Például fizika tantárgy ugyanúgy létezik az informatikus hallgatóknak, mint az építészeknek, de a tananyag különböző. Ez a struktúra ezt a helyzetet tökéletesen lekezeli. Létre kell hozni a fizika tárgyat a Műszaki Informatika alkategóriájaként és az Építészmérnök szak alkategóriájaként is, ám amikor publikálunk egy jegyzetet az informatikusoknak, akkor a Műszaki Informatika „*Fizika”* alkategóriáját pipáljuk be, ha pedig az építészeknek, akkor pedig az ő „*Fizika”* alkategóriájukat jelöljük be.
Hallgatói szemszögből is tökéletesen átlátható így a rendszer. Ha én informatikus vagyok és szükségem van fizika jegyzetre, akkor természetesen én a Műszaki Informatika alá tartozó Fizika alkategóriára fogok kattintani, és ekkor csak a nekem szóló fizikajegyzetekkel fogok találkozni.
Miután létrehoztuk az összes kategóriát, kattintsunk a *Beállítások/Írás* menüpontra, majd állítsuk be a bejegyzések alapértelmezett kategóriának a „*Hírek*”-et.

5.4. Saját sablon

A WordPress-ben a sablonok egyszerű HTML fájlok. A dinamikus adatokat php függvényhívások segítségével érhetjük el. A kimenet formátumát, rendezését és szűrését a függvények paramétereinek módosításával tudjuk formázni. A kimenet nem tartalmaz semmilyen HTML formázó entitást, kivéve a linkeket és a listákat.
A kimenetek grafikai megjelenését nekünk kell a függvényhívás körül HTML formázás segítségével megvalósítani. A sablonfájlokban található HTML kódok ajánlás szerint nem tartalmazhatnak grafikai formázást, csupán a struktúra definiálására szabad őket használni. A megjelenésért felelős kódokat egy külső CSS fájlban kell definiálnunk.
Elmondható tehát, hogy az oldalak strukturális felépítését a sablonunk PHP fájljai tartalmazzák, a grafikai megjelenést pedig a CSS fájl határozza meg.

5.4.1. Sablonok logikai felépítése

A logikailag összetartozó területeket a redundancia és az áttekinthetőség elkerülése érdekében külön-külön fájlokban egyetlen egyszer kell definiálni. Így alapesetben a következő fájljaink vannak:

- header.php: a fejléc megjelenéséért felelős
- footer.php: a lábléc megjelenéséért felelős
- sidebar.php: az oldalsáv megjelenéséért felelős
- comments.php: a bejegyzésekhez és oldalakhoz tartozó megjegyzések megjelenéséért felelős
- index.php: a főoldal megjelenéséért felelős. Itt van beágyazva a *header.php, sidebar.php, footer.php.*
- single.php: egyetlen bejegyzés megjelenéséért felelős. Itt van beágyazva a *header.php, sidebar.php, footer.php, comments.php.*
- hasonlóképpen a kisebb logikai egységek is önálló fájlt kaptak (pl.: *searchform.php*)

5.4.2. Kategóriák listázásának módosítása

Alapesetben a WodPress kategórialistázó eljárása hierarchia nélkül kilistázza a sidebar-on azokat a kategóriákat, amelyekbe született már bejegyzés. A kategóriák létrehozásakor azonban én egy jól strukturált rendszert készítettem, hogy jól átlátható módon elkülönüljenek a szakok és a kurzusok. Emellett létrehoztam néhány technikai jellegű kategóriát, amiket nem feltétlenül szeretnék kiíratni. Ebben az esetben a listázó függvény paramétereit kell módosítani.
Nyissuk meg *sidebar.php* fájlt, és keressük meg az alábbi kódrészletet:

```
<?php
wp_list_cats('sort_column=name&optioncount=1&hierarchical=0');
?>
```

Cseréljük le a következőre:

```
<?php
wp_list_cats('sort_column=name&optioncount=1&hierarchical=1&chil
d_of=18');
?>
```

Paraméterek magyarázata:

hierarchical= (érték: 0 vagy 1)

Ha értéke 0, akkor nem veszi figyelembe listázáskor a kategóriák alárendeltségét. Ha értéke 1, akkor igen, és az alkategóriákat kicsivel beljebb írja ki, mint a szülőjét.

child_of= (érték: egész)

Ha ennek a paraméternek nincs értéke, akkor minden kategóriát kilistáz. Ha értéket adunk neki, akkor csak a megadott azonosítójú kategória alkategóriáit listázza ki. Nekünk pont erre van szükségünk. Nálam a *Jegyzetek* kategória azonosítója 18, így ezt az értéket adva csak a szakok és az az azokhoz tartozó tantárgyak listázódnak ki. Ezért kellett anno így létrehozni a kategóriákat. A *Jegyzetek* kategória csupán ez a szűrési lehetőség miatt kellett.

5.4.3. A WordPress Loop

Egyetlen egy része van a sablonfájloknak, amik nem egyetlen függvényhívásként működnek, ez pedig az úgynevezett WordPress Loop. Ez a rész felelős a bejegyzések listázásáért. Ez egy vezérlési szerkezetekből és függvényhívásokból álló kódrészlet. Mivel első ránézésre bonyolultnak tűnik, ezért az átlag felhasználók nem is nyúlnak hozzá. Viszont ezt is lehet paraméterezni segédfüggvényekkel, így sokkal egyedibbé szabhatjuk oldalunk működését.

Alapesetben ez a kódrészlet így néz ki:

```
<?php if (have_posts()) : ?>
<?php while (have_posts()) : the_post(); ?>
```

```
<div class="post" id="post-<?php the_ID(); ?>">
<h2>
<a href="<?php the_permalink() ?>" rel="bookmark" title="<?php
printf(__('Permanent Link to %s','default.i18n'),
the_title('','',false)); ?>"><?php the_title(); ?>
</a>
</h2>
<small>
<?php the_time(__('F jS, Y','default.i18n')) ?> <!-- by <?php
the_author() ?> -->
</small>
<div class="entry">
<?php the_content(__('Read the rest of this entry
&raquo;','default.i18n')); ?>
</div>
<p class="postmetadata">
<?php printf(__('Posted in %s','default.i18n'),
get_the_category_list(', ')); ?> | <?php
edit_post_link(__('Edit','default.i18n'), '', ' | '); ?>  <?php
comments_popup_link(__('No Comments &#187;','default.i18n'), __('1
Comment &#187;','default.i18n'), __('% Comments
&#187;','default.i18n')); ?>
</p>
</div>
<?php endwhile; ?>
```

Látható, hogy az algoritmus előbb megvizsgálja, hogy létezik-e egyáltalán bejegyzés (`have_posts()`függvényhívás), majd elkezdi őket listázni egy while ciklusban. Az is látható, hogy az egy bejegyzéshez tartozó elemek mindegyik egy függvényhívás segítségével jelenik meg (bejegyzés címe: `the_title()`, dátum: `the_time(__('F jS, Y','default.i18n'))`, bejegyzés kategóriája: `get_the_category_list(', ')`), stb…), így ezen függvényeket átrendezve, paramétereiket módosítva vagy törölve megváltozik a bejegyzés kinézete is.

Az oldalba készítek egy kiemelt híreket tartalmazó dobozt. Ez úgy fog működni, hogy az oldalsávra teszek egy WordPress Loop-ot úgy, hogy az csak a „*hírek*” kategóriába írt üzeneteket listázza ki. Igen ám, de a főoldalon lévő Loop is listázná ezeket a bejegyzéseket, hiszen arra nincsen semmilyen korlátozás beállítva, így ezek a hírek két helyen is megjelennének. Ez így nem lenne egy okos megoldás. A Loop-nak paraméterként megadhatjuk azt, hogy csak kizárólag egy adott kategória bejegyzéseit listázza, és egy beépített függvényhívással el tudjuk érni, hogy bizonyos azonosítójú kategóriába publikált bejegyzések ne kerüljenek bele a ciklusba. Ezért a főoldalon lévő Loop-ot is módosítanunk kell, méghozzá úgy, hogy a bejegyzést csak akkor jelenítse meg, ha annak kategóriája nem a „*hírek*”. Ezt az *in_category()* függvénnyel érhetjük el úgy, hogy paraméterének a „*Hírek*” kategória azonosítóját adjuk meg (nálam ez 17).
A Loop második sora után szúrjuk be a következő kódot.

```
<?php if (in_category('17')) continue; ?>
```

Ez a kód egy elágazás a Loop-on belül. Ha az *in_category()* függvény igazzal tér vissza, akkor a ciklus továbblép a következő bejegyzés kiértékelésére.

5.4.4. Az oldalsáv többi részének módosítása

Ha módosítottuk a főoldalban lévő Loop-ot, akkor most következhet az oldalsávba történő Loop létrehozása. Ez úgy fog működni, hogy csak a „*Hírek*” kategóriába publikált bejegyzéseket fogja listázni. Azok közül is csak a legfrissebb hármat.
Nyissuk meg a *sidebar.php* fájlt, és illesszük be az alábbi kódot oda, ahol szeretnénk, hogy a hírek megjelenjenek. Én rögtön a keresés alá tettem:

```
<?php query_posts('category_name=hirek&showposts=3'); ?>
  <?php while (have_posts()) : the_post(); ?>
    <div class="kiemelthirek">
    <div class="kiemelthirek-cim"><?php the_title(); ?></div>
    <div class="kiemelthirek-datum"><?php the_time('Y. F j -
H:i') ?> <?php edit_post_link('Szerkeszt','',''); ?></div>
    <?php the_content('<br />Tovább');?>
```

```
    <div class="kiemelthirek-megjegyzesek">
    <?php comments_popup_link('0 megjegyzés', '1 megjegyzés', '%
megjegyzés'); ?></div>
    </div>
<?php endwhile; ?>
```

A Loop működését a *query_posts()* függvénnyel és annak paramétereivel tudjuk módosítani. Jelen esetben két paramétert használtam:

category_name (érték: string)

Ezzel a paraméterrel adhatjuk meg, hogy csak a „Hírek” kategória bejegyzései jelenjenek meg. Látható, hogy a kódban nem „*Hírek*” van írva, hanem „*hirek*”. Ez azért van, mert minden kategória rendelkezik egy „szépített névvel” is (ez a címsorbeli név), amit az eredetiből generál a WordPress úgy, hogy azok csak kisbetűket tartalmazzanak, és ne szerepeljenek benne hosszú, illetve speciális karakterek.

showposts (érték: egész)

Ezzel a paraméterrel pedig azt határozhatjuk meg, hogy az aktuális Loop hány elemet jelenítsen meg.

Látható, hogy a megjelenítendő objektumok egyedi stílust kaptak. A *style.css* fájlba a következő kódot illesszük:

```
.kiemelthirek {
      background-color: #6AACE6;
      margin-bottom: 20px;
      padding: 5px;
      border: 2px solid #4283B9;
      color: #FFFFFF;
}
.kiemelthirek-cim {
      font-size: 12px;
      font-weight: bold;
}
```

```
.kiemelthirek-datum {
	color: #E1F0FF;
	border-bottom-width: 1px;
	border-bottom-style: dotted;
	border-bottom-color: #FFFFFF;
	padding-top: 3px;
	padding-bottom: 3px;
}
.kiemelthirek-megjegyzesek {
	text-align: right;
}
.kiemelthirek a {
	color: #FFFFFF;
}
```

5.5. Plugin programozása

A WordPress tartalomkezelő rendszerhez rengeteg beépülő modul (plugin) érhető el az interneten. Ezen kiegészítések segítségével számos extra funkcióval ruházhatjuk fel oldalunkat. A legtöbb plugint a hivatalos WordPress oldalon találhatjuk (http://wordpress.org/extend/plugins). Előfordulhat az is, hogy annyira egyedi kiegészítőre van szükségünk, amit még nem készített el egyetlen fejlesztő sem. Ebben az esetben magunknak kell megírnunk a kívánt alkalmazást. Ehhez a feladathoz már elkél egy kis programozói tudás.
Ebben a fejezetben azt mutatom be, hogy hogyan kell olyan plugint készíteni, amely az API-n keresztül szerves része tud lenni rendszerünknek.

5.5.1. Elnevezési szokások, elhelyezkedés, kötelező elemek

Korábban már volt szó a pluginek helyéről. Mint már említettem a kiegészítőket a *wp-content/plugins* mappába kell helyeznünk. A kiegészítő lehet egyetlen fájl, de több állományból is állhat, ekkor egy almappát kell létrehozni a számára. A fájlok

elnevezésekor arra kell ügyelnünk, hogy a név egyértelműen utaljon a plugin nevére, de az csak az angol abc kis betűit és az aláhúzás ”_” karaktert tartalmazhatja.
Ha készítünk egy plugint, akkor annak az elejére még két fontos dolgot szokás beilleszteni. Ez a fejrész és a licensz.
A fejrész olyan információkat tartalmaz a programunkról, amit a WordPress kiolvas és azt jeleníti meg az elérhető pluginek listája közt. Íme egy tipikus fejrész:

```
<?php
/*
Plugin Name: A plugin neve
Plugin URI: A plugin internetes elérhetősége (pl http://az-en-
pluginom.hu)
Description: A plugin rövid leírása
Version: Verziószám (pl 1.0)
Author: Készítő neve
Author URI: A készítő weboldala (pl http://mondovics.extra.hu)
*/
?>
```

A második szokásos elem, amit a plugin elejére szoktak illeszteni az a licensz. Mivel a WordPress a GNU GPL licensz alatt áll, ezért az ahhoz készült kiegészítőket is ezzel kell ellátni.

```
<?php
/*  Copyright YEAR  PLUGIN_AUTHOR_NAME  (email : PLUGIN AUTHOR
EMAIL)

    This program is free software; you can redistribute it
and/or modify it under the terms of the GNU General Public
License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

    This program is distributed in the hope that it will be
useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
```

```
PURPOSE.  See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public
License along with this program; if not, write to the Free
Software Foundation, Inc., 59 Temple Place, Suite 330, Boston,
MA  02111-1307  USA
*/
?>
```

Ezután kezdhetjük írni az utasításainkat. Figyeljünk arra, hogy mind a fejrész, mind pedig a licensz egy kikommentezett része a fájlnak. A tényleges programkód előtt nem lehet semmilyen kimenete a php fájlunknak (még egy szóköz sem).

5.5.2. Hurkok

A pluginek úgynevezett hurkok segítségével kapcsolódhatnak a WordPress-hez. Ezek olyan előre definiált függvények, amelyek különböző eseményekhez lehetnek rendelve. Ha egy plugin hozzá van rendelve egy eseményhez egy adott hurkon keresztül, akkor a WordPress működése megváltozik, és a vezérlés az adott pluginra adódik át. Például ha egy plugin kapcsolódik egy „*filter*” típusú hurokhoz, aminek a „*the_title*” a neve, akkor a bejegyzések adatbázisból való letöltődésük után a cím mező kezelése átadódik a pluginnak, és az a benne meghatározott módon megváltoztatja a bejegyzés címét.

5.5.3. Adatok mentése az adatbázisba

Ha komolyabb plugint készítünk, akkor az valószínűleg rendelkezik némi konfigurációs lehetőségekkel. Ezen beálltásokat is egy WordPress hurok segítségével tudjuk regisztrálni illetve elérni.

Opció rögzítése:

```
add_option($name, $value, $description, $autoload);
```

A függvény paraméterei a következők:

- **$name**: Sztring, kötelező megadni. Az opció neve.
- **$value**: Sztring, opcionális. Az opcióhoz tartozó érték.
- **$description**: Sztring, opcionális. Az opció rövid ismertetése.
- **$autoload**: Opcionális, értéke ’yes’ vagy ’no’. Alapértelmezésben ’yes’, ekkor ez az opció automatikusan átadódik a *get_alloptions* függvénynek.

Opció lekérdezése:

```
get_option($option);
```

A függvény paraméterei a következők:

- **$option**: Sztring, kötelező megadni. A lekérdezni kívánt opció neve.

Opció felülírása:

```
update_option($option_name, $newvalue);
```

A függvény paraméterei a következők:

- **$option_name**: Sztring, kötelező megadni. A módosítani kívánt opció neve.
- **$newvalue**: Sztring, kötelező megadni. Az opció új értéke.

5.5.4. Plugin hozzákapcsolása az admin menühöz

Ha a plugin rendelkezik saját opciókkal, akkor azokat célszerű egy önálló menüpontban megjeleníteni az admin felületen. Szintén hurkok segítségével tudunk a menühöz saját tartalmat kapcsolni.

Menüpont regisztrálása:

```
add_options_page(page_title, menu_title, access_level/capability,
file, [function]);
```

A függvény paraméterei a következők:

- **Page_title**: oldal címe
- **Menu_title**: menü címe

- **access_level/capability**: jogosultsági szint ami felett a menü megjelenik
- **file**: annak a fájlnak a neve, amelyikben a menüponthoz tartozó oldal tartalma van
- **function**: annak a függvények a neve, amelyik a menüponthoz tartozó oldal tartalmát jeleníti meg

Menüpont betöltése a menüsorba:

```
add_action('admin_menu', $function);
```

$function: az a függvény, amelyikben a menüpont regisztrálásáért felelős hurok van.

5.5.5. Megjegyzés cenzúrázó plugin

A WordPress rendelkezik egy beépített funkcióval, amellyel lehetőségünk van az oldalra érkező megjegyzéseket moderálni. Ez úgy működik, hogy megadjuk a tiltandó szavakat, és ha az adott megjegyzés ezek valamelyikét tartalmazza, akkor az egész hozzászólás moderálásra kerül.
A plugin készítésének bemutatása céljából én ennél egy engedékenyebb kiegészítőt fogok elkészíteni. Ez csak a tiltott szavakat fogja törölni vagy helyettesíteni, a megjegyzés többi részét pedig érintetlenül hagyja.

Először is regisztráljuk a „*Beállítások*” almenüi közé a plugin beállításait tartalmazó menüt.

```
function cenzura_admin_menu(){
    add_options_page('Cenzúra', 'Cenzúra', 10, 'cenzura_admin',
'cenzura_admin');
}
add_action('admin_menu', 'cenzura_admin_menu');
```

Létrehozzuk a plugin beállításait tartalmazó rekordokat az adatbázisban.

```
function cenzura_admin(){
    add_option('cenzura_tiltott_szavak', '', 'Ezek a szavak
lesznek tiltva.');
    add_option('cenzura_helyettesito_szo', '[cenzúrázva]', 'Ez
jelenik meg a tiltott szavak helyett.');
```

Ez a kód mindig lefut, de az *add_option()* függvény úgy működik, hogy ha az adott opció már létezik, akkor azt nem módosítja.
Ezután definiáljuk magát a megjelenítendő oldalt. Első lépésként lekérjük az adatbázisból a tárolt beállításokat és egy-egy változóban (`$db_helyettesito_szo`, `$db_tiltott_szavak`) rögzítjük azokat. Később ezt adjuk át a beviteli mezők kezdőértékének.

```
<?php
    $db_helyettesito_szo = get_option('cenzura_helyettesito_szo');
    $db_tiltott_szavak = get_option('cenzura_tiltott_szavak');
    ?>
    <div class='wrap'>
    <h2>Megjegyzések cenzúrázása</h2>
    <form method='post' action='<?php echo
'?page=cenzura_admin&op=cenzura_save' ?>'>
    <input style='margin-top:10px; margin-bottom:10px;'
type='submit' value='Beállítások mentése' />
    <div>
    <p>Összegyűjtöttünk néhány gyakran tiltandó szót. A
feketelistát <a href="http://aeonline.hu/?p=454">innen</a>
töltheted le.</p>
    </div>
    Tiltott szavak: (szóközzel elválasztva, kisbetű/nagybetű
mindegy)<br />
    <textarea style='width: 99%; height: 150px;'
name='tiltott_szavak'>
<?php echo $db_tiltott_szavak; ?> </textarea><br /><br />
    A tiltott szavak legyenek helyettesítve ezzel:<br />
    <input type='text' name='helyettesito_szo' value='<?php echo
$db_helyettesito_szo; ?>' />
```

```
    <br />
    <input style='margin-top:10px; margin-bottom:10px;'
type='submit' value='Beállítások mentése' />
    </form>
    </div>
```

8. ábra: A megjegyzés cenzúrázó plugin beállításait tartalmazó oldal

Amennyiben módosítunk a beállításokon az új adatokat el kell menteni az adatbázisba. A változtatásról egy GET metódussal átadott változón keresztül értesül a program.

```
<?php
if($_GET['op']==cenzura_save)
        {
        update_option('cenzura_tiltott_szavak',
$_POST['tiltott_szavak']);
        update_option('cenzura_helyettesito_szo',
$_POST['helyettesito_szo']);
```

```
        // "adatok mentve" üzenet
        ?>
        <div style='background: red; margin: 10px; padding:
10px; color: white;'>Adatok mentve</div>
        <?php
        }
?>
```

Végül a helyettesítést végző függvény következik. Első lépésben lekérjük az adatbázisból a tiltott szavakat tartalmazó sztringet, és a szóközök mentén tömbbé alakítjuk. Ezután a helyettesítő szót is lekérdezzük, majd az eredményeket eltároljuk egy-egy változóban.

```
function cenzura($text){
    $tiltott_szavak = get_option('cenzura_tiltott_szavak');
    $tiltott_szavak = explode(" ", $tiltott_szavak);
    $helyettesito_szo = get_option('cenzura_helyettesito_szo');
```

A hatékonyság érdekében a cserének érzéketlennek kell lennie a kis- és nagybetűkre. Az *str_ireplace()* függvény kiváló lenne erre a célra, ám ez a függvény csak a PHP5-től érhető el. Gondolni kell tehát a régebbi verziójú PHP környezetet futtató szerverekre is. Szerencsére a PHP online kézikönyvének ezen függvényéhez kapcsolódó hozzászólások közt van egy programkód, amely tömbökön végzett betűnagyságra érzéketlen keresést és cserét valósít meg.

```
function make_pattern(&$pat, $key) {
       $pat = '/'.preg_quote($pat, '/').'/i';
   }
   if(!function_exists('str_ireplace')){
       function str_ireplace($search, $replace, $subject){
           if(is_array($search)){
               array_walk($search, 'make_pattern');
           }
           else{
               $search = '/'.preg_quote($search, '/').'/i';
```

```
            }
            return preg_replace($search, $replace, $subject);
        }
    }
```

Most már csak el kell végezni a paraméterként megadott sztringen a helyettesítést, és vissza kell adni a megszűrt információt.

```
$text = str_ireplace($tiltott_szavak, $helyettesito_szo, $text);
return $text;
```

Végül a lényeg: a megjegyzésekre egy szűrő típusú hurkon keresztül rácsatolom a plugint a WordPress-re.

```
add_filter('comment_text', 'cenzura');
```

6. Összegzés

Néhány éve már természetes, hogy az internet bárki számára elérhető. Már nem számít különlegességnek az, hogy bárki passzív használója legyen a világhálónak. Ennek köszönhetően az aktív részvétel is egyre jobban előtérbe kerül. Ma már nem csak használjuk az internetet, nem csak „tartalmat fogyasztunk”, hanem aktív részesei vagyunk a tartalom előállításának. A családi fotókat és videókat online közösségi hálózatokon publikáljuk. Ezen dolgok tükrében már természetes az, hogy minden szervezetnek és vállalkozásnak van saját weboldala. Az ingyenes tartalomkezelő rendszerek használata azon cégek, szervezetek vagy egyéni felhasználók számára teremt költséghatékony megoldást, akik nem tudnak megfizetni egy saját fejlesztésű portálmotort.

Nem kellett sokat gondolkoznom, hogy szakdolgozatomban ezen terület lehetőségeit mutatom be. Választásom azért esett a WordPress rendszerére, mert egy ideje használom már saját weboldalam működtetéséhez, és ez idő alatt nagyon

megkedveltem. Hihetetlenül egyszerű működtetni, és az alap feladatok ellátására tökéletesen alkalmas. A WordPress egy blogmotor, de mi is a blog tulajdonképpen? Egy olyan platform, ahol publikációkat oszthatunk meg és rendszerezhetjük azokat, valamint kapcsolatot teremthetünk felhasználóink és olvasóink között. Ennek tükrében nagyon sok weboldalra rámondhatjuk, hogy az egy blog, ezért a WordPress kiváló választás lehet. Egy hallgatói információs rendszer is publikációkat (jegyzeteket, híreket) oszt meg, és a hallgatókkal való kapcsolatteremtést tűzi ki célul (publikációk körül kialakuló beszélgetés a megjegyzések révén).

A WordPress előnye egyszerűségében és hatalmas felhasználói táborában mutatkozik meg. Szakdolgozatomból látható, hogy az alap rendszer beüzemelése hihetetlenül egyszerű, de könnyen kibővíthető, hiszen több ezer sablon és majd tízezer plugin érhető el bárki számára ingyenesen az interneten. Ha ez nem lenne elég, akkor némi programozói ismeretek segítségével pedig könnyen készíthetünk saját sablont, vagy írhatunk egyéni igényeinkhez alkalmazkodó plugint.

A másik érv, hogy a WordPress tartalomkezelőről írjam szakdolgozatomat az volt, hogy magyar nyelvű részletes leírás még nem nagyon található a rendszerről, ezért remélem, hogy dolgozatommal hozzájárulok ezen remek rendszer népszerűsítéséhez.

Dolgozatomban bemutatott rendszer bárki számára megtekinthető a http://pte-jegyzetek.extra.hu címen.

7. Irodalomjegyzék

[1] Matt Zandstra: Tanuljuk meg a PHP4 használatát 24 óra alatt, Budapest, Kiskapu, 2001

WordPress tartalomkezelővel kapcsolatos leírások, segédletek:

[2] http://www.word-press.hu

[3] http://codex.wordpress.org/Writing_a_Plugin

PHP programozással kapcsolatos segédletek:

[4] http://hu.php.net