

SZAKDOLGOZAT

Göbl Ádám
2012

WEBALKALMAZÁS FEJLESZTÉSE PHP KERETRENDSZER SEGÍTSÉGÉVEL

Pécsi Tudományegyetem
Természettudományi Kar
Informatika Tanszék
Pécs, 2012



Témavezető, belső konzulens:
Rébay Viktor
Tanársegéd

Készítette:
Göbl Ádám
Programtervező-informatikus Bsc.
Nappali tagozat
gobl.adi@gmail.com

HALLGATÓI NYILATKOZAT

Alulírott diplomázó hallgató kijelentem, hogy jelen szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Egyéb jelentős segítséget nem vettem igénybe.

Az elkészült szakdolgozatban található eredményeket a Pécsi Tudományegyetem, mint a feladatot kiíró intézmény, saját céljaira térítés nélkül felhasználhatja.

Kelt: Pécs, 2012

.....

hallgató aláírása

FELADATLAP

szakdolgozat készítéséhez

A hallgató neve: Göbl Ádám

A dolgozat címe: Webalkalmazás fejlesztése PHP keretrendszer segítségével

Belső konzulens neve: Rébay Viktor

beosztása: tanársegéd

A feladat leírása:

Egy működő példán keresztül mutatom be, hogy milyen előnyei és hátrányai vannak webalkalmazás készítése során, ha PHP keretrendszer segítségével készül a program. Részletesen bemutatom a Yii Framework keretrendszert, ismertetem a MVC (model-view-controll) tervezési mintát, majd mindezt a gyakorlatban is alkalmazom az elkészülő rendszer fejlesztése során. Az elkészülő rendszer egy grafikai munkákban érdekelt vállalat megrendeléseit kezeli és 2 fő modulból áll: Az egyik modulban az ügyfelek feladhatják és nyomon követhetik rendeléseiket, a másik modulban a cég alkalmazottai az adatbázis alapján a rendelkezésre álló kellékekből (papírok, nyomdai eszközök, stb.) és szolgáltatásokból állítják össze a megrendelés tételes leírását.

Beadási határidő: 2012. 05. 14.

.....

konzulens

.....

szakfelelős

Konzulensek ellenőrzési időpontjai:

| Belső konzulens | | Külső konzulens | |
|-----------------|--------------------|-----------------|--------------------|
| Dátum | Konzulens aláírása | Dátum | Konzulens aláírása |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

A szakdolgozat beadható.

20hónap

.....

belső konzulens

TARTALOMJEGYZÉK

| | |
|--|----|
| Előszó | 6 |
| 1. Bevezetés és célkitűzések | 6 |
| 2. A Yii framework bemutatása | 9 |
| 2.1. Miért a Yii-t választottam? | 9 |
| 2.2. A Yii Framework-ről általában | 11 |
| 2.3. A Yii Framework jellemzői..... | 13 |
| 2.3.1. Model-View-Controller (MVC) tervezési minta [16]..... | 13 |
| 2.3.2. Front Controller tervezési minta | 17 |
| 2.4. A Yii Framework telepítése | 18 |
| 2.5. A Yii Framework felépítése | 20 |
| 2.6. Yii konvenciók | 22 |
| 2.7. Bootstrap | 25 |
| 3. Webalkalmazás tervezése | 27 |
| 3.1. SQL és MySQL | 27 |
| 3.2. Fejlesztői eszközök | 28 |
| 3.2.1. Netbeans | 28 |
| 3.2.2. SQLYog | 30 |
| 3.2.3. Firebug | 31 |
| 3.2.4. Test Mail Server Tool | 32 |
| 3.3. Verziókövetés | 33 |
| 3.4. Használati eset diagram | 37 |
| 3.5. Az adatmodell tervezése | 38 |
| 3.5.1. Felhasználók tárolása | 39 |
| 3.5.2. Városok és irányítószámok | 41 |
| 3.5.3. Felhasználható eszközök | 42 |

| | |
|---|----|
| 3.5.4. Megrendelések nyilvántartása | 42 |
| 3.5.5. A teljes adatmodell | 44 |
| 4. A webalkalmazás elkészítése | 45 |
| 4.1. Kódgenerálás | 45 |
| 4.1.1. A user modell bemutatása | 47 |
| 4.1.2. A user controller bemutatása | 48 |
| 4.1.3. A user view-k bemutatása | 49 |
| 4.2. ExtJS Javascript Library | 50 |
| 5. A webalkalmazás működés közben | 52 |
| 5.1. Az internetes megrendelés felülete | 52 |
| 5.2. Az adminisztrációs modul | 58 |
| Összegzés | 66 |
| IRODALOMJEGYZÉK | 67 |
| Mellékletek | 70 |
| Telepítési útmutató | 71 |

ELŐSZÓ

Engem már általános iskolás koromban is foglalkoztatott a programozás, mióta egy számítástechnika szakkörön a tanárom bevezetett a QBasic [1] világába. Lelkesen írogattam a különböző programokat, amik általában abból álltak, hogy bizonyos bementekre különböző válaszokat írtak ki a képernyőre, esetleg rajzoltak is valamit. Ezekre ráunva később megismertem a Pascal [2] nyelvet, amivel már több dolgot tudtam megvalósítani. Arra lettem figyelmes, hogy egy-egy problémát különböző nyelveken más és más módokon lehet megközelíteni, és megoldani.

Évekkel később a főiskolán folytatódott a tendencia, különböző programozási nyelveken hasonló algoritmusokat írtunk meg, oldottunk meg ugyanolyan problémákat.

Szakedolgozatom témája ezekre a hasonlóságokra épül: egy általam korábban elkészített webalkalmazást készítettem el újból, ezúttal PHP keretrendszer segítségével, és ezen a példán keresztül mutatom be, hogy milyen előnyei és hátrányai vannak, ha keretrendszerrel dolgozunk.

1. BEVEZETÉS ÉS CÉLKITŰZÉSEK

A 2000-es évek elején két vállalkozó szellemű barát a fő foglalkozásuk mellett indítottak egy nyomdaipari vállalkozást, hívjuk őket a továbbiakban „cég”-nek. Eleinte csak kisebb megrendeléseket teljesítettek, nagyobb megbízásuk még nem igen volt. Mivel az akkori ügyfelek nagyon elégedettek voltak velük, hírük ment a városban, és egyre több munkájuk lett. Akkoriban egy kockás füzetben vezették, hogy mikor mit rendeltek tőlük, az mennyibe került, mennyi nyereségük volt, stb. A vállalkozás kezdett beindulni, így hamar rájöttek, hogy ha hosszútávon sikeresek akarnak lenni, valamilyen jobb módszert kell használniuk a dokumentálásra, hogy később könnyen visszakereshető legyen egy-egy munka, valamint az ügyfelek adatai se tűnjenek el. Úgy gondolták, egyelőre jó lesz nekik, ha a Microsoft Access adatbázis-kezelő programban összeállítanak egy adatbázist, és ott le tudják dokumentálni a szükséges adatokat, később pedig könnyen vissza tudják keresni azokat.

Ez a megoldás egy ideig elég volt számukra, de így nehézkes volt megoldani, hogy mindketten lássák az adatokat, tudják azokat szerkeszteni vagy újakat felvinni. Ahogy

egyre szélesedett az internet felhasználók köre Magyarországon, ők is abban látták a megoldást, hogy egy interneten keresztül elérhető alkalmazásra lenne szükségük.

| Év | Felhasználók | Népesség | Népesség % |
|------|--------------|------------|------------|
| 2000 | 715.000 | 10.174.853 | 7.0% |
| 2007 | 3.500.000 | 10.037.768 | 34.9% |
| 2010 | 6.176.400 | 9.992.339 | 61.8% |

1. ábra

Internetet használók száma alakulása Magyarországon [3]

Az új nyilvántartó eszközük megírására (megíratására) 2008-ban szánták rá magukat, és ekkor jöttem én a képbe.

Az igényeik a következők voltak:

- Webes alkalmazás, PHP alapokon, MySQL adatbázissal
- A korábbi programjuk adatbázisának újragondolása, átalakítása, a régi adatok migrálása az új adatszerkezetbe
- Felhasználónévvel és az ahhoz tartozó jelszóval azonosított beléptetés különböző jogkörökkel

Legyen benne:

- Ügyfélnyilvántartó modul és ahhoz tartozó kezelőfelület
- Megrendelés nyilvántartó felület
- Árajánlat adó modul, ahol össze tudják állítani a megrendelések teljesítéséhez szükséges eszközöket
- Az eszközök menedzselését lehetővé tevő felület

Az alkalmazást elkészítettem a fent megjelölt igények alapján, amit azóta is használnak. Az elmúlt két évben php-programozással foglalkoztam, és minél több tapasztalatot szereztem, annál jobban megláttam a fent említett alkalmazás hiányosságait, illetve, hogy hogy lehetne még jobb, optimálisabb kódot készíteni. Az alkalmazás használói elégedettek a mostani programjukkal, nem működik lassan, mindketten tudják használni, tehát az ő igényeiket kielégíti.

Ahogy az előszóban már említettem, régóta foglalkoztat a programozással kapcsolatban, hogy egy-egy problémát többféleképpen lehet megközelíteni, megoldani. A cég

anonimitása érdekében a jelenleg működő alkalmazást nem tudom bemutatni, viszont úgy döntöttem, hogy a szakdolgozatomban bemutatni kívánt keretrendszerhez ideális alany lenne.

Szakdolgozatom célja, hogy a bevezető elején említett nyomdaipari vállalkozás alkalmazását újraírva bemutattam, hogy milyen előnyei és hátrányai vannak, ha PHP keretrendszer segítségével készül a program.

Az új alkalmazás célkitűzései:

- teljesen objektum-orientált legyen
- megvalósítsa az MVC (model-view-controll) tervezési mintát
- optimálisabb adatszerkezete legyen
- kihasználja a keretrendszer szolgáltatásait

Két fő részből áll:

- Internetes felület, ami teljesen publikus (tulajdonképpen a cég honlapja). Itt van lehetősége az ügyfeleknek regisztrálni, majd regisztrált felhasználóként belépve fel tudják adni megrendeléseiket, nyomon tudják követni a korábbiakat.
- Adminisztrátori felület, ahol a cég munkatársai tudják kezelni a megrendeléseket, össze tudják állítani az árajánlatot, tudják menedzselni az eszközöket. Az adminisztrátori felület megjelenítéséért az ExtJS Javascript library felel.

2. A YII FRAMEWORK BEMUTATÁSA

2.1. Miért a Yii-t választottam?

A szakdolgozatom írásakor az interneten több mint 140 webalkalmazás készítő keretrendszer [4] érhető el, ezek közül kb. 40 PHP-s keretrendszer. Több szempontból is össze vannak hasonlítva, pl. hogy hányas PHP verziótól elérhetők, van-e beépített tesztelési lehetőség, támogatja-e az Ajax-ot és ha igen, akkor milyen technológiát használ ehhez, stb. Ezen adatok alapján már valamennyire körvonalazódik, hogy kinek mire van szüksége, de még így is sok lehetőség van választani. Az látszik, hogy nem könnyű objektív döntést hozni, hiszen ha különböző angol vagy magyar nyelvű fórumokon érdeklődünk, azt fogják javasolni, hogy próbáljunk ki 2-3 keretrendszert az ismertebbek közül, és válasszunk az alapján.

Nagyon sok ilyen témájú cikk jelent meg az utóbbi időben, hogy melyik a legjobb PHP keretrendszer. Egy 2011-es cikk [5] szerint az 5 legjobb PHP keretrendszer sorrendben a következő:

1. Yii Framework
2. Code Igniter
3. Zend Framework
4. Kohana PHP Framework
5. Cake PHP

| PHP Framework | PHP4 | PHP5 | MVC | Multiple DB's | ORM | DB Objects | Templates | Caching | Validation | Ajax |
|---------------|------|------|-----|---------------|-----|------------|-----------|---------|------------|------|
| Akelos | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ash.MVC | - | ✓ | ✓ | - | - | ✓ | ✓ | - | ✓ | - |
| CakePHP | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ |
| CodeIgniter | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | - |
| DIY | - | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | - | ✓ |
| eZ Components | - | ✓ | - | ✓ | - | ✓ | ✓ | ✓ | ✓ | - |
| Fusebox | ✓ | ✓ | ✓ | ✓ | - | - | - | ✓ | - | ✓ |
| PHP on TRAX | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | ✓ |
| PHPDevShell | - | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PhpOpenbiz | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ |
| Prado | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| QPHP | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | - | ✓ | ✓ |
| Seagull | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Symfony | - | ✓ | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ |
| WACT | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | - | ✓ | - |
| WASP | - | ✓ | ✓ | - | - | ✓ | ✓ | - | ✓ | ✓ |
| Yii | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Zend | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ZooP | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | ✓ |

2. ábra

A legnépszerűbb PHP keretrendszerek összehasonlítása [6]

Mindent összevetve én az alábbi 5 szempont figyelembe vétele után döntöttem a Yii Framework mellett:

- Milyen méretű alkalmazást fogok fejleszteni?
- Szükségem van-e speciálisabb komponensekre?
- Milyen a keretrendszer dokumentáltsága, támogatottsága?
- Milyen időközönként adnak ki frissítéseket, javításokat a keretrendszerhez?
- Mennyire „kötött” a keretrendszer?

Az első kérdés talán a legfontosabb, hiszen itt dől el, hogy egyáltalán szükség van-e arra, hogy keretrendszerrel fejlesszünk. Mivel a PHP egy HTML-be ágyazott, egyszerű script nyelv, bizonyos bonyolultságig a keretrendszer használatának több hátránya van, mint előnye. Tény, hogy bármilyen keretrendszert használunk is, a sok betöltendő állomány miatt az ilyen weblapok lassabban fognak működni, és nagyobb erőforrás-igényűek, mint egyszerűbb társaik. Ez a lassúság a mai korszerű technológiai háttér és jó sávszélesség mellett kevésbé érzékelhető. A Yii Framework a PHP keretrendszerek között az egyik leggyorsabb, erről a következő fejezetben írok bővebben.

Szintén fontos szempont lehet a keretrendszer kiválasztásakor, hogy milyen komponenseket tartalmaz alapból, valamint hogy szükségünk lesz-e egyáltalán ilyenekre? A Yii esetében – mint azt látni fogjuk – szinte mindenre találunk beépített vagy letölthető [7] komponenst, sőt, saját magunk is készíthetünk, ha úgy adódik.

Ahhoz, hogy kihasználhassuk a keretrendszer nyújtotta előnyöket, elengedhetetlen a megfelelő dokumentáció. A Yii ilyen téren elég jól áll, hiszen a honlapjukról számos példát tölthetünk le, maga a Yii telepítő is példákkal van csomagolva, ill. letölthető egy ún. „Yii appliance”, ami egy virtuális gépen (VMware) futó szerver, és tartalmazza a Yii-t néhány bemutató alkalmazással. Szintén a honlapjukon található egy API dokumentáció (ami sajnos nélkülözi a példákat), valamint néhány tutorialt [8] is találhatunk a letöltések között. Ezek mellett a Yii támogatottsága miatt sok fórumot, és néhány könyvet is találhatunk az interneten, bár ezek többsége angolul van.

Azt is érdemes figyelembe venni, hogy az adott keretrendszert milyen fejlesztőcsapat fejleszti, milyen időközönként adnak ki frissítéseket, a jelzett hibákat javítják-e. A Yii nyílt forráskódú, ráadásul a Github-on [9] (online verziókövető) nyomon követheti bárki, hogy mikor milyen frissítést eszközöltek a fejlesztők. Általában naponta több

változtatás (*commit*) is bekerül, amiket néhány havonta adnak ki stabil verzióként. Eddig átlagosan kb. 3 havonta adtak ki új verziót, de az épp aktuális javításokat az előbb említett Github-ról bárki letöltheti.

Az utolsó kérdésre adott válasz nagyban függ attól, hogy az adott keretrendszer mennyi automatizációt tartalmaz, mennyire szabható testre, mennyire kötelezi a programozót a komponensek használatára, stb. A Yii-ben van automatikus kódgenerálás (a Gii komponensről később bővebben is írok), de saját magunk is megírhatunk bármit, nem kötelez bennünket a komponensek használatára, tetszőlegesen használhatjuk a PHP nyelv összes lehetőségét.

2.2. A Yii Framework-ről általában

A Yii Framework egy webes alkalmazásokhoz készített általános keretrendszer, ami teljesen objektum-orientált, és implementálja a széles körben elterjedt MVC [10] (Model-View-Controller) tervezési mintát. Nincsenek benne forradalmi újdonságok, a PHP web fejlesztés eddigi tapasztalatait összegzi oly módon, hogy tervezői a népszerű, széles körben használt és elismert keretrendszerekből merítettek ötleteket, összeválogatva azok legjobb megoldásait, és egy új, gyors rendszert hoztak létre.

A Yii fejlesztése 2008 januárjában indult hivatalosan, készítője a Yii Software LCC csapat. A fő fejlesztő Xiang Que (USA), az ő munkáját egy nemzetközi összetételű csapat segíti (a dolgozat írásának idejében 6-an voltak rajta kívül [11]). Érdekességgént jegyezném meg, hogy magyar vonatkozása is van a Yii Framework-nek, ugyanis Beregszászi István is a fejlesztő csapat tagja volt egy ideig (2009. szeptembertől 2010. márciusig), valamint a hivatalos fórumon is van magyar nyelvű topik.

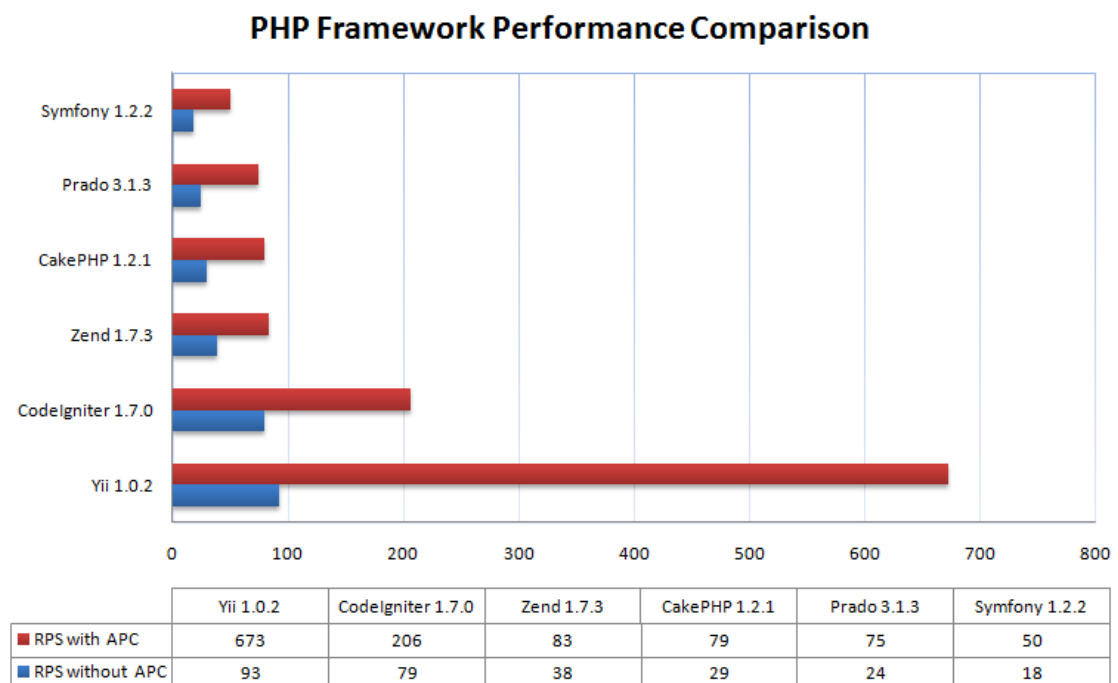
2008. december 3-án adták ki a fejlesztők az 1.0.0-ás stabil és nyilvánosságnak szánt verziót [12], a szakdolgozat írásának idejében pedig az 1.1.10-es verziónál tartanak (a szakdolgozatban az 1.1.8-as verziót használtam, az alkalmazás írásának kezdetekor még az volt a legfrissebb).

A név jelentése: „*Yes, it is!*”. A fejlesztők szerint a nevet kiejtve (Yeee) jelképezi a Yii tulajdonságait: *easy*, *efficient*, *extensible*, azaz könnyű, hatékony és bővíthető. Ezekre a tulajdonságokra a későbbiekben még visszatérek, hogy a gyakorlatban én hogy látom, megvalósulnak-e.

A készítő a keretrendszerüket szabadsoftverként hozzák forgalomba, BSD licenc [13] alatt. „A BSD licenc, ellentétben a GNU GPL-lel, gyakorlatilag bármit megenged a termék bármely felhasználójának vagy fejlesztőnek. Ebbe beletartozik a termék (szoftver) átírása és zárt (bináris) formában való terjesztése, kereskedelmi forgalmazása. Mindössze olyan alapvető kitételek szerepelnek benne, mint hogy az eredeti fejlesztők nem vállalnak semmilyen felelősséget a termék használatával kapcsolatban”. [14]

Mint már fentebb említettem, a Yii-ben nincsenek forradalmi újdonságok, más keretrendszerek legjobb megoldásait vették át és valósították meg egy keretrendszerben. Ezek a források a következők: Prado (mint fő forrás), Ruby on Rails, jQuery (alapból támogatja), Symfony, Joomla.

A saját honlapján olvasható bemutatkozó szerint „a Yii egy ingyenes, szabad-felhasználású PHP keretrendszer, ami PHP 5-ben íródott és jól alkalmazható web alkalmazások készítéséhez. Leredukálja a fejlesztési időt, elősegíti a kód újrafelhasználhatóságát azáltal, hogy könnyű, hatékony és rugalmas” [11]. A gyorsaságát alátámasztandó, találhatunk egy diagramot, amin a sebességét hasonlították össze más keretrendszerekkel.



3. ábra
Kérések teljesítésének összehasonlítása [15]

Miért ilyen gyors a Yii? A honlapjukon olvasható leírás szerint azért, mert „a Yii alaposan alkalmazza a lazy loading technikát. Például addig nem include-olja a különböző osztályokat a fájlokba, amíg azokat nem használjuk ténylegesen, valamint nem példányosít objektumokat, csak az első használat után.” [15]

2.3. A Yii Framework jellemzői

2.3.1. Model-View-Controller (MVC) tervezési minta [16]

A Yii Framework a hangsúlyt az objektum orientáltságra helyezi. Mint már fentebb említettem, a széles körben elterjedt MVC (magyarul: modell-nézet-vezérlés) tervezési mintát implementálja, teljesen elválasztva egymástól a megjelenítést, az adatokat és működést. A *model* (modell) réteg jelképezi az adatokat és az azok közti logikát, a *view* (nézet) a felhasználói felületet (formok), és kimenteket (HTML, XML, JSON) foglalja magában, míg a *controller* (vezérlő) e két réteg közti kommunikációt vezérli. Például ha az alkalmazásunkat egy böngészőben és egy hordozható eszközön is futtatni szeretnénk, akkor csak új nézetekre van szükségünk.

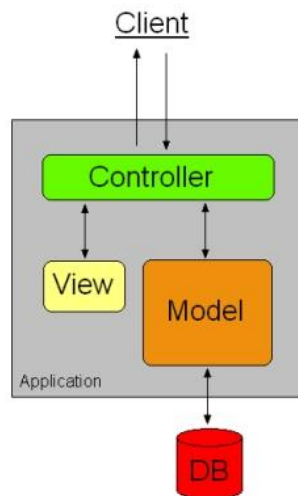
A mintát Trygve Reenskaug írta le először 1979-ben, miután a Smalltalkon dolgozott a Xerox kutatói laboránál. Az eredeti megvalósítás részletesen a nagy hatású *Applications Programming in Smalltalk-80: How to use Model-View-Controller* [17] című tanulmányban olvasható.

Gyakori egy alkalmazás több rétegre való felbontása: megjelenítés (felhasználói felület), tartománylogika és adatelérés. Az MVC-ben a megjelenítés tovább bomlik nézetre és vezérlőre. Az MVC sokkal inkább meghatározza egy alkalmazás szerkezetét, mint az egy programtervezési mintára jellemző.

Mivel a webes alkalmazások esetében a kód pont erre a három részre osztható, ezért nagyon elterjedt ez a minta közöttük. A Yii esetében (és a legtöbb keretrendszerénél is így van) az MVC minta meghatározza a rendszer mappaszerkezetét is. Ez a szeparáltság eleinte megszokást igényel attól a fejlesztőtől, aki korábban nem használta ezt a mintát, de később nagy gyakorlati haszna van: egységes lesz tőle az alkalmazás, valamint hibakeresésnél vagy új funkció írásánál egyértelműen tudni fogjuk, hogy a kód mely részéhez kell nyúlnunk.

„Habár az MVC-nek sok értelmezése létezik, a vezérlés menete általánosságban a következőképp működik:

1. A felhasználó valamilyen hatást gyakorol a felhasználói felületre (pl. megnyom egy gombot).
2. A vezérlő átveszi a bejövő eseményt a felhasználói felülettől, gyakran egy bejegyzett eseménykezelő vagy visszahívás útján.
3. A vezérlő kapcsolatot teremt a modellel, esetleg frissíti azt a felhasználó tevékenységének megfelelő módon (pl. a vezérlő frissíti a felhasználó kosarát). Az összetett vezérlőket gyakran alakítják ki az utasítás mintának megfelelően, a műveletek egységbezárásáért és a bővítés egyszerűsítéséért.
4. A nézet (közvetve) a modell alapján megfelelő felhasználói felületet hoz létre (pl. a nézet hozza létre a kosár tartalmát felsoroló képernyőt). A nézet a modellből nyeri az adatait. A modellnek nincs közvetlen tudomása a nézetről.
5. A felhasználói felület újabb eseményre vár, mely az elejéről kezdi a kört.” [16]



4. ábra

Az MVC tervezési minta egy általános reprezentációja [16]

Modell: „Az alkalmazás által kezelt információk tartomány-specifikus ábrázolása. A tartománylogika jelentést ad a puszta adatnak.” [16]

Sok alkalmazás használ állandó tároló eljárásokat (pl. adatbázis) adatok tárolásához. Az MVC nem említi külön az adatelérési réteget, mert ezt beleérti a modellbe.

Yii framework esetében gyakorlatilag táblánként 1 osztály implementációját jelenti a modell elkészítése. A modell osztályban meg kell adnunk az adatbázis tábla nevét, az elsődleges kulcsot, és további adatokat is itt definiálhatunk: kapcsolatok (*relations*), szabályok (*rules*), ellenőrzéseket (*validator*) stb.

A Yii modell osztály a *CModel* osztályból származtatott osztály, mely egy adategységet foglal magában, ami lehet egy adatbázis tábla egy sora, vagy egy, a felhasználó által küldött beviteli adatok halmaza.

A Yii a modelleket tehát kétféle csoportba sorolja:

- Űrlap modell (*CFormModel*): felhasználói bemenetből származó adatok, melyek kezelés után nem tárolódnak. Ilyen például egy bejelentkező űrlap: a felhasználónév és jelszó ebben az esetben nem tárolandó adat, csak addig van rá szükség, amíg a felhasználót hitelesítjük.
- Aktív rekord (*CActiveRecord*): a modell egy adatbázis tábla egy sorát képviseli, melynek mezői az osztály változói, a megszokott CRUD (Create, Read, Update, Delete) [18] műveletek implementálva vannak benne.

Ezzel a megoldással egyszerűen hozhatunk létre új rekordokat, például így:

```
$article = new Article;  
  
$article -> title="Új cikk címe";  
  
$article -> content="Cikk tartalma";  
  
$article -> save();
```

View: „Megjeleníti a modellt egy megfelelő alakban, mely alkalmas a felhasználói interakcióra, jellemzően egy felhasználói felületi elem képében, ilyenek például az űrlapok, menük, táblázatok stb. Különböző célokra különböző nézetek létezhetnek ugyanahhoz a modellhez.” [16] A Yii esetében a view-k általában egyszerű PHP scriptek, ahol a megjelenítési logikát kell megírni. A view neve megegyezik a PHP script fájl nevével, melyet a `protected/views/ControllerID` mappában kell elhelyezni, és a nézet megjelenítéséhez a `render` metódust kell hívni (erről később részletesebben is szót ejtek). A view script-ben a `$this` paranccsal a hívó Controller-t érjük el, további változókat a `render` metódusban adhatunk át, a következő módon:


```

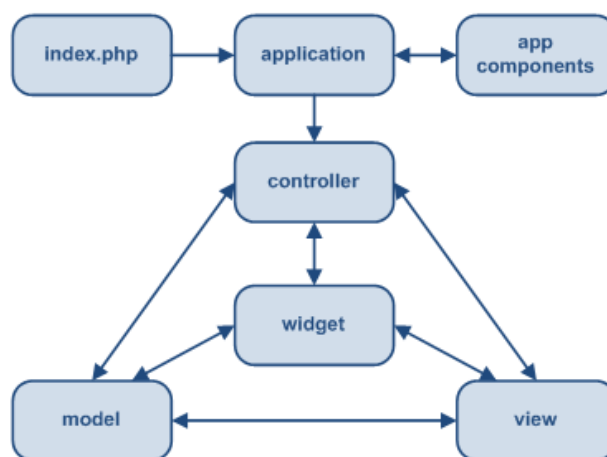
$this->render('edit', array(
    'var1'=>$value1,
    'var2'=>$value2,
));

```

A fejlesztéskor egyedül itt találkozhatunk a HTML nyelvvel. Alapértelmezésként használhatjuk a Yii által felkínált sablonozó rendszert, de akár más ilyen eszközt is használhatunk. A szakdolgozatomban elkészített alkalmazás adminisztrátor moduljában megjelenítésnek az ExtJS-t használom.

A view-k mellett kell megemlítenem a *layout*-okat is, amik tulajdonképpen speciális view scriptek, és a több oldalon ismétlődő elemeket foglalják magukban. A gyakorlatban ez azt jelenti, hogy egy alap layout-ot hozunk létre, amiben deklaráljuk a *doctype*-ot, behúzzuk az alap CSS-eket, Javascript-eket, és a layout-on belül jelenítjük majd meg a különböző view-kat. Természetesen több layout-unk is lehet.

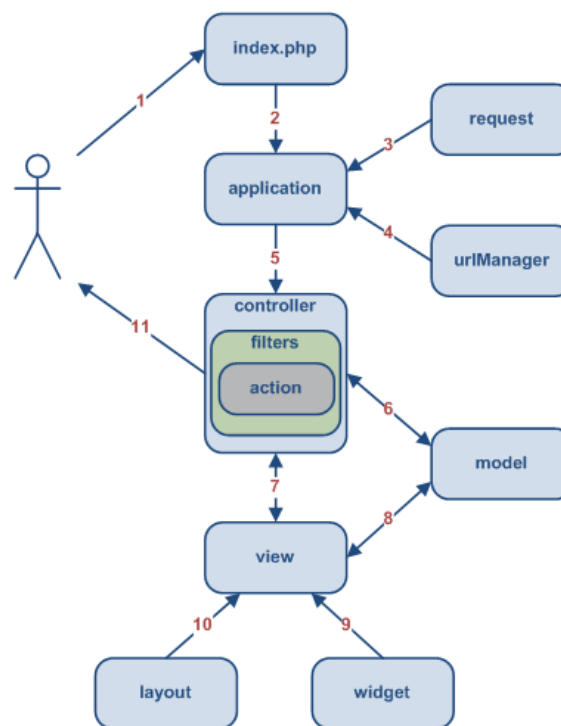
Controller: Az egész mintát a *controller* köti össze egy nagy egységgé. Az eseményeket, jellemzően felhasználói műveleteket dolgozza fel és válaszol rájuk, illetve a modellben történő változásokat is kiválthat. Yii esetében (és sok más keretrendszerénél is) a controller-ek osztályok formájában jelennek meg és ezen osztályok metódusai az ún. action-ök. Az 5. ábra mutatja be, hogy valósul meg az MVC minta Yii esetében.



5. ábra
A Yii Framework felépítése [19]

2.3.2. Front Controller tervezési minta

A Yii Framework a Front Controller mintát is implementálja, amely a kérelmek kezeléséért felelős: a felhasználótól érkező kérelmet feldolgozza, mely alapján meghatározza a megfelelő controller osztályt, amelynek átírányítja a kérést további feldolgozásra. Ahhoz, hogy részletesebben megértsük a Front Controller mintát, nézzük meg, mi történik a HTTP kérésekkel a *routing* és *dispatching* során [19]. A 6. ábra részletesen bemutatja ennek megvalósulását.



6. ábra

Kérelem kezelése Yii Frameworkben [19]

Például, a felhasználó a következő címet írja be a böngészőjébe: `http://www.example.com/index.php?r=post/show&id=1`, akkor a fenti ábra lépései szerint a következők történnek:

1. A web szerver kezeli a kérelmet, futtatja az `index.php` belépő scriptet
2. A belépő script létrehozza a FC *singleton* (egyke) osztályt
3. A FC kinyeri a kérelemből a részletes információkat a *CHttpRequest* komponens segítségével
4. A *CUrlManager* komponens segítségével meghatározza a megfelelő controller-t és az action-t. A példában a `post` a `PostController` osztályra utal, a `show` a

showAction nevű metódusára, melyet egyértelműen meghatároz a controller osztály

5. Az FC példányosítja a kért controller osztályt, létrehozza és futtatja a controller-hez tartozó szűrőket (pl. hozzáférés szabályozás) és ha ezek engedik, futtatja az action-t
6. Az action kiolvassa az 1-es azonosítójú Post modellt az adatbázisból
7. Az action összeállítja a show nevű nézetet a lekért Post modell alapján
8. A view megjeleníti a Post modell attribútumait
9. A view futtathat *widget*-eket (komplex, de önálló, újrahasznosítható nézetelemek)
10. A view összeállítja az eredményt a layout-ba ágyazva
11. Az action befejezi a view készítését, és megjeleníti azt a felhasználónak

2.4. A Yii Framework telepítése

Az alábbi lépéseket követve tudjuk telepíteni a Yii Framework-öt:

1. A hivatalos honlapról letölthetjük tar-gzip vagy zip által tömörített változatban valamelyik verziót, célszerű a legfrissebbet (bár hamarosan kiadják a 2.0-ás verziót, ami csak PHP 5.3-tól elérhető, erre nem árt figyelni). Használatához szükséges egy webszerver, minimum 5.1.0-ás PHP-val. A fejlesztők az Apache alatt teszteltek és ezt is ajánlják, a szakdolgozathoz én is ezt használtam (2.2.21).
2. Miután kicsomagoltuk az állományokat a webszerverünk által elérhető könyvtárba, három könyvtárat kapunk és néhány fájlt. Javasolt ellenőrizni, hogy teljesülnek-e a feltételek a webszerverünkön a Yii futtatásához, ezt a következőképp tesztelhetjük: nyissuk meg a böngészőnkben a `<webszerver>/yii/requirements/index.php`-t. Ennek egy lehetséges kimenetét látjuk a 7. ábrán.

| Name | Result | Required By | Memo |
|------------------------------------|---------|---|--|
| PHP version | Passed | Yii Framework | PHP 5.1.0 or higher is required. |
| \$_SERVER variable | Passed | Yii Framework | |
| Reflection extension | Passed | Yii Framework | |
| PCRE extension | Passed | Yii Framework | |
| SPL extension | Passed | Yii Framework | |
| DOM extension | Passed | CHtmlPurifier , CWsdlGenerator | |
| PDO extension | Passed | All DB-related classes | |
| PDO SQLite extension | Warning | All DB-related classes | This is required if you are using SQLite database. |
| PDO MySQL extension | Passed | All DB-related classes | This is required if you are using MySQL database. |
| PDO PostgreSQL extension | Passed | All DB-related classes | This is required if you are using PostgreSQL database. |
| Memcache extension | Passed | CMemCache | |
| APC extension | Warning | CApcCache | |
| Mcrypt extension | Passed | CSecurityManager | This is required by encrypt and decrypt methods. |
| SOAP extension | Passed | CWebService , CWebServiceAction | |
| GD extension with FreeType support | Passed | CCaptchaAction | |
| Ctype extension | Passed | CDateFormatter , CDateTimeParser , CTextHighlighter , CHtmlPurifier | |

■ passed
 ■ failed
 ■ warning

Apache/2.2.21 (Win32) mod_ssl/2.2.21 OpenSSL/0.9.8r PHP/5.2.14 [Yii Framework/1.1.8](#) 2012-05-01 14:27

7. ábra
Yii Framework követelmények tesztelése

3. Ezután a Yiic nevű kódgenerálóval létrehozhatunk parancssorból egy alkalmazás vázat (*yii skeleton application*), ezt a következőképp tehetjük meg:
 - a. Nyissunk meg egy parancsértelmezőt
 - b. Navigáljunk abba a könyvtárba, ahova a Yii fájljait másoltuk
 - c. Futtassuk le az alábbi parancsot: `yiic webapp <elérési út>/alkalmazasneve`
 - d. Ezzel elindítjuk a Yiic kódgenerálót, ami ha a megerősítő kérdésre igen-t mondunk, a megadott elérési út alá létrehozza az alkalmazás fájljait

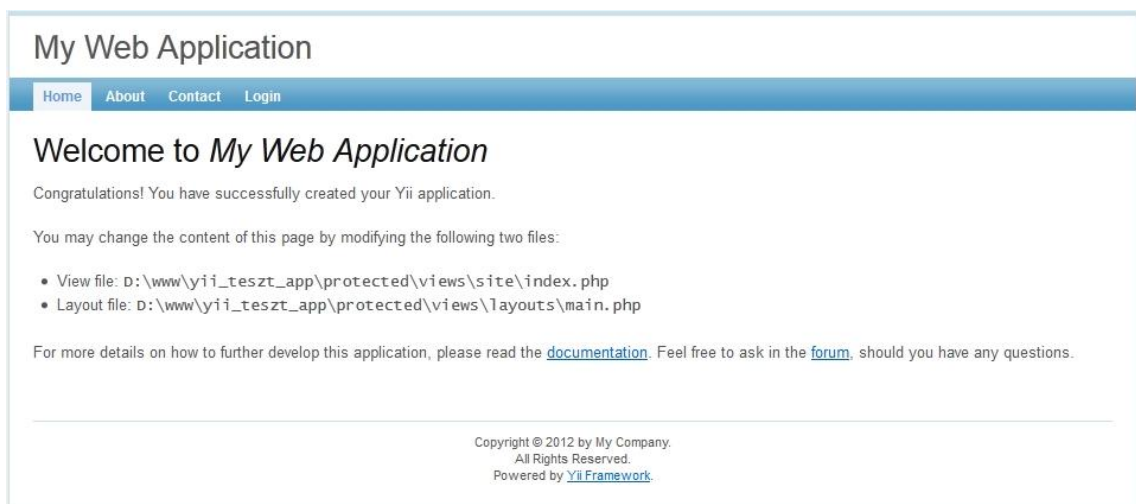
```

C:\Windows\System32\cmd.exe
D:\www\yii_1_8>framework\yiic webapp d:\www\yii_teszt_app
Create a Web application under 'D:\www\yii_teszt_app'? [Yes!No] yes
mkdir D:\www\yii_teszt_app
mkdir D:\www\yii_teszt_app/assets
mkdir D:\www\yii_teszt_app/css
generate css/bg.gif
generate css/form.css
generate css/ie.css
generate css/main.css
generate css/print.css
generate css/screen.css
mkdir D:\www\yii_teszt_app/images
generate index-test.php
generate index.php
mkdir D:\www\yii_teszt_app\protected
generate protected/.htaccess
mkdir D:\www\yii_teszt_app\protected\commands
mkdir D:\www\yii_teszt_app\protected\commands\shell
mkdir D:\www\yii_teszt_app\protected\components
generate protected/components/Controller.php
generate protected/components/UserIdentity.php
mkdir D:\www\yii_teszt_app\protected\config
generate protected/config/console.php
generate protected/config/main.php
generate protected/config/test.php
  
```

8. ábra
Yiic tool létrehozza a skeleton application fájljait

A Yii tool sajnos nem működik kielégítően. Windows és Linux alatt is a parancs fájl a PHP értelmezőt próbálja meghívni, de ha a PHP értelmező könyvtára nincs az elérési utak között, akkor hibát fog jelezni a Yii tool. Kétféle megoldás van erre: vagy a fenti parancs elé be kell szúrni az *útvonal/php* kiegészítést, vagy operációs rendszertől függően be kell állítani a PHP elérési útvonalát. [20]

4. Ha a parancs sikeresen lefutott, akkor a megadott helyen elkészült az alkalmazás váza (*skeleton application*), amit a böngészőnkön keresztül megtekinthetünk, ha a következő címet hívjuk: `<webszerver>/alkalmazasneve`. A létrehozott alkalmazás mérete a sok Javascript-nek köszönhetően majdnem 1MB. A 9. ábrán láthatjuk az elkészült alkalmazás nyitólapját. Ennek az alkalmazásnak az alapján már könnyen elindulhatunk, hogy megírjuk a saját alkalmazásunkat.



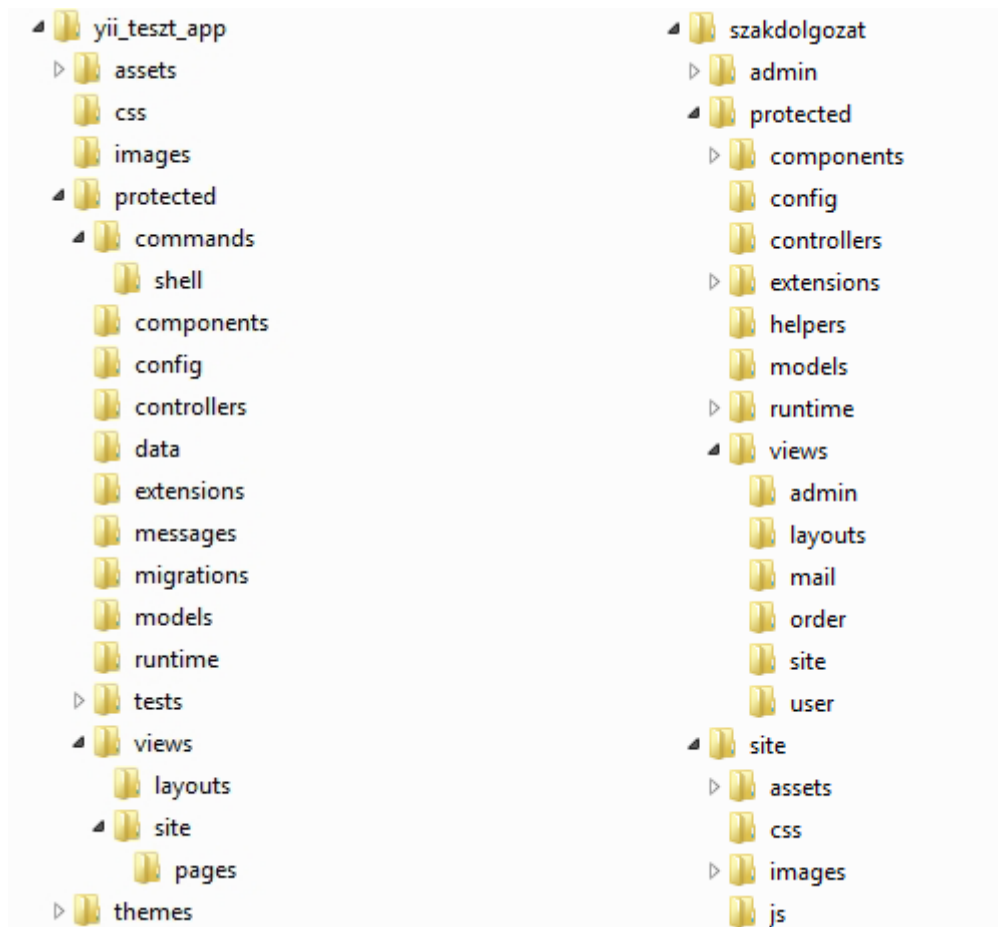
9. ábra

Yii tool által létrehozott alkalmazás kezdőlapja

2.5. A Yii Framework felépítése

Ebben a fejezetben ismertetem a Yii Framework mappastruktúráját, összehasonlítva az előző fejezetben létrehozott *skeleton* alkalmazás struktúráját a saját alkalmazásoméval, bemutatva ezzel, hogy hogyan lehet rugalmasan kezelni a keretrendszert, ugyanis a saját alkalmazásomnál néhány mappára nem volt szükségem, illetve egy-két esetben máshol helyeztem el bizonyos állományokat (CSS, képek), mint ahol a mintaalkalmazásban található.

A korábban említett MVC tervezési minta meghatározó szerepű az alkalmazás struktúrájában is, meghatározza a program alapvető fájljainak elhelyezkedését. Az alapértelmezett felépítés és a saját alkalmazásom felépítését a 10. ábra szemlélteti.



10. ábra

Yii skeleton application mappaszerkezete (bal oldal), saját alkalmazás mappaszerkezete (jobb oldal)

A skeleton appliacion mappáinak tartalma:

yii_teszt_app/

index.php

assets/

css/

images/

protected/

commands/

components/

config/

controllers/

data/

extensions/

A webalkalmazás belépési pontja

Közös fájlok helye

Stílus lapok

Képek

Az alkalmazás védett fájljai

yiic parancsok

Belső alkalmazások, portletek

Konfigurációs fájlok

Vezérlők (kontrollerek)

Adatfájlok, pl. SQL dump

Kiegészítések

| | |
|-------------|----------------------------|
| messages/ | Nyelvi fájlok |
| migrations/ | Adatbázis migrációs fájlok |
| models/ | Model fájlok |
| runtime/ | Ideiglenes fájlok |
| tests/ | Teszt szkriptek |
| views/ | Nézethez szükséges fájlok |
| layouts/ | Különböző layout-ok |
| themes/ | Témák |

Az alkalmazás egyetlen belépési pontja a `<webszerver>/alkalmazasneve/index.php`, ez az ún. *bootstrap* fájl (erről részletesebben a 2.7. fejezetben írok). Az alkalmazás védett fájljai a *protected* könyvtárban vannak (models, views, controllers, config), ezeket kívülről nem lehet más módon elérni. Az alapvető beállításokat a *protected/config/main.php* tartalmazza. Maga a Yii Framework az alkalmazáson kívül helyezkedik el, elérési útját a belépési fájlban kell megadnunk, így egy szerveren több alkalmazás is tudja használni közösen a keretrendszer fájljait.

Az én alkalmazásomnak ezek közül nincs szüksége a következőkre: *commands* (mert nem használom benne a Yii tool-t), *data* (az adatbázis dump-ot máshol mellékelem), *messages* (az alkalmazás csak egynyelvű), *migrations* (nincsenek migrációs fájlok), *tests* (nincsenek teszt szkriptek) és *themes* (nincsen több megjelenítési téma).

Az *admin* mappában van az adminisztrációs modul, ez egy külön alkalmazás, külön belépő (index.php) szkripttel. Az *admin* mappában található a megjelenítésért felelős ExtJS Javascript library fájljai is (ExtJS rövid bemutatása a 4.2-es fejezetben).

A *site* mappában helyeztem el azokat az állományokat, amiket kívülről is el lehet érni, tehát a képek, Javascript-ek, stíluslapok itt találhatók, valamint az alkalmazás belépési pontja is.

A *protected* mappában vannak az alkalmazás védett fájljai, ahogy a *skeleton application* esetében is.

2.6. Yii konvenciók

Általában amikor keretrendszert használunk fejlesztéshez, az egy nagyobb projektet feltételez, vagy azt, hogy nem csak egy fejlesztő dolgozik az adott projekten. Esetemben most ugyan egyik állítás sem helytálló, mégis érdemes betartani bizonyos kódolási stílusokat annak érdekében, hogy jobb, átláthatóbb kódot írassunk, és egyszerűbben el

tudjuk kerülni a hibákat. Ha egységes kódolási stílust alkalmazunk egy projektben, később könnyebb hozzányúlni, vagy hibát keresni. A következőkben ismertetem a Yii Framework által ajánlott és használt konvenciókat [21]. A szakdolgozatom során igyekeztem a Zend által javasolt kódolási stílust alkalmazni [22], ha az nem ütközött a Yii által ajánlottal.

URL: Alapértelmezettként, a Yii a következő URL-sémát használja:

```
http://hostname/index.php?r=ControllerID/ActionID
```

A controllerek-nek és a controller-en belüli action metódusoknak egyedi azonosítójuk van, ez alapján kapja meg az index.php az „r” (*route*) paraméterét. Ha nem adunk meg action-t, akkor a controller alapértelmezett action metódusát hívja, ami alaphoz az *index* nevű action, de ez konfigurálható a *defaultAction* változóval. Természetesen ún. *SEO friendly* (keresőbarát) URL-eket is használhatunk.

Kódolás: A Yii az ún. „camelCase” elnevezést ajánlja változók, függvények, metódusok és osztályok elnevezésére, amikor minden szó kezdőbetűjét nagybetűvel írjuk, és a szavakat szünet nélkül írjuk egymás után. A változók (pl. *\$basePath*) és függvények (pl. *runController()*) kezdőbetűjét érdemes kisbetűvel kezdeni, hogy meglehessen különböztetni az osztály nevektől (pl. *LinkPager*). A *private* láthatósági szinttel rendelkező változókat és függvényeket alulvonás karakterrel kezdjük írni (pl. *\$_actionList*). Fontos szabály, hogy a vezérlő osztályok elnevezése a „Controller” szóval kell, hogy végződjön. Így a vezérlő azonosítója az osztály nevének a „Controller” szó előtti része lesz. Például a *PageController* osztály azonosítója a *page*.

Konfiguráció: Az alkalmazás konfigurációját PHP szkriptként kell megírunk, amelyek tömb formájában adják vissza a konfigurációs beállításokat.

```
return array(  
    'param1' => 'value1',  
    'param2' => 'value2'  
    ...  
);
```

Konfiguráció elérése az alkalmazásból:

```
Yii::app() -> param1
```


Nagyobb alkalmazás esetében érdemes szétbontani a konfigurációt több fájlba, melyekben résztömböket adunk vissza. Az én alkalmazásom esetében 4 részre bontottam a konfigurációkat:

- custom – ebben az olyan beállítások vannak, amik serverenként eltérhetnek egymástól (pl. adatbázis elérés, cache-elés, stb.)
- admin – az admin modul beállításai
- main – az alapbeállítások
- site – az oldal működésével kapcsolatos beállítások

Fájl: A fájlok elnevezése és használata a típusuktól függ.

Az osztályokat tartalmazó fájlokat arról az osztályról kell elnevezni, amit *public* osztályként tartalmaznak. Egy osztályban egy *public* láthatósági szintű osztály lehet. Például, a *PageController.php*-ban a *PageController* osztály van publikusként definiálva. *Private* láthatósági szintű osztályokat abba a fájlba kell tennünk, ahol az őt használó *public* osztály van.

A megjelenítésért felelős *view* fájlokat a nézet nevééről kell elneveznünk, és a vezérlő azonosítójáról elnevezett mappában kell lennie. Például, a felhasználói regisztrációt a *UserController* kezeli. Ahhoz, hogy megjelenítsük a regisztrációs űrlapot, a *UserController.php*-ban hívjuk az alábbi parancsot:

```
$this->render('regisztracio');
```

Ekkor a keretrendszer a *regisztracio.php*-t fogja megjeleníteni, ami a *views/user* mappában van. Itt a *view* neve „regisztracio”.

Mappák: A fájlokat javasolt a Yii által ajánlott struktúrát követve elhelyezni, de bármelyiket felülbírállhatjuk a konfigurációs fájlokban. A Yii által ajánlott mappaszerkezet a következő:

- Webroot/protected: itt kell lenniük a védett PHP szkripteknek és adatfájloknak, ezt a mappát nem lehet kívülről elérni, csak az alkalmazáson keresztül. A *CWebApplication::basePath* változóban lehet felülírni a helyét.
- Webroot/protected/runtime: ebben a mappában vannak az alkalmazás futása során létrejött ideiglenes fájlok. A webszervernek írási jogosultságot kell adni erre a mappára, különben nem fog tudni futni az alkalmazás. A *CApplication::runtimePath* változóban lehet felülírni a helyét.

- Webroot/protected/modules: ha modulokra van osztva az alkalmazás, akkor a különböző modulokat tartalmazza.
- Webroot/protected/controllers: a vezérlő osztályokat tartalmazza. A *CWebApplication::controllerPath* változóban lehet megváltoztatni.
- Webroot/protected/views: a megjelenítésért felelős fájlok találhatók itt mappákba szervezve. *CWebApplication::viewPath* változóval tudjuk átírni a helyét.
- Webroot/protected/views/ControllerID: Erről már volt szó a fájlok elnevezésénél: egy vezérlőhöz tartozó megjelenítő fájlokat tartalmazza. A *CController::viewPath* változóban lehet felülírni.
- Webroot/protected/views/layouts: Ebben a mappában vannak a *layout* fájlok. Ennek az elérési útját a *CWebApplication::systemViewPath* változóban állíthatjuk át.

Adatbázis: az alábbi szabályokat kell betartanunk az adatbázisoknál, ha Yii-vel fejlesztünk:

- Az adatbázis táblák nevei és a mezők neve csak kisbetűket tartalmazhatnak
- A szavak elválasztására alulvonást kell használni (*például: ez_a_tabla_neve*)
- Tábla neveknél következetesen vagy egyes-, vagy többes számot használjunk. A Yii Framework – ahogy általában a komolyabb adatmodellezéssel kapcsolatos szakirodalmak is – az egyes számot javasolja, én is ehhez tartottam magam.
- Érdekes a táblák neve elé valamely prefixet használni, így ha egy adatbázisban több alkalmazás táblái vannak, könnyű lesz őket megkülönböztetni. Én a „sz_” prefixet használtam a tábláim előtt.

2.7. Bootstrap

Ahogy a mappaszerkezetnél már szó volt róla, az alkalmazásnak csak egy belépési pontja van, a felhasználók csak ezt a fájlt érik el közvetlenül. Angolul *bootstrap*-nek hívják ezt a belépési pontot. Mit is jelent a *bootstrap*? Lefordítva nem sok értelme van a szónak, az angol terminológia ezzel a szóval írja le azt az eljárást, amikor a kódot inicializáljuk, létrehozuk az adatbázis kapcsolatot, betöltjük a konfigurációs fájlokat, stb. Ha belegondolunk, hogy mi a különbség egy HTTP protokollon futó webalkalmazás, és egy asztali alkalmazás között, akkor láthatjuk, hogy míg az asztali

alkalmazás folyamatosan fut, addig a HTTP gyakorlatilag kérés-válasz alapú, állapotmentes protokoll: amint visszajött a válasz, a program nem fut tovább. Ha újra jön egy kérés a szerver felé, jön a válasz, majd ismét leáll a működés. Látszik, hogy vannak olyan dolgok, amikre minden egyes oldalbetöltéskor szükség van. Egy keretrendszerrel megírt alkalmazás esetén ilyen például a keretrendszer inicializálása, a jogosultságkezelés, az adatbázis-kapcsolat, stb.

Nézzük meg, hogy is néz ki a belépő szkript:

```
//beállítjuk a Yii Framework elérési útját
$yii = dirname(__FILE__) . '/../..../yii_1_8/framework/yii.php';
//beállítjuk az alapvető konfigurációs fájlt
$config = dirname(__FILE__) . '/../protected/config/site.php';
//fejlesztés során debug-módban használjuk az alkalmazást
defined('YII_DEBUG') or define('YII_DEBUG',true);
require_once($yii);
//futtatjuk az alkalmazást
Yii::createWebApplication($config)->run();
```

Szintén a mappaszerkezetnél volt szó az alkalmazás védett fájljairól, amik a *protected* mappában vannak. Ezeket az állományokat csak az alkalmazáson keresztül lehet elérni, meg kell védenünk őket attól, hogy külső helyről meg lehessen hívni őket. Apache webszerver esetén a *.htaccess*-ben a *deny from all* paranccsal tehetjük meg ezt. A szakdolgozatom esetében pedig a *vhosts* konfigurációs állományban tettem ezt meg, részletesebben erről a telepítési útmutatóban írok.

3. WEBALKALMAZÁS TERVEZÉSE

Most, hogy ismertettem a Yii Framework tulajdonságait, a bevezetőben említett alkalmazás elkészítésének lépéseit mutatom be. Általánosságban elmondható, hogy a jó és eredményes munkához jó eszközök kellenek. Így van ez a szoftverfejlesztéssel is, a megfelelő eszközök használatával könnyebbé, gyorsabbá, jobbá tehetjük az elkészült munkát. A továbbiakban röviden ismertetem, hogy milyen eszközöket vettem igénybe a fejlesztés során.

3.1. SQL és MySQL

Az elkészült alkalmazás az adatokat adatbázisban tárolja. „Adatbázisokra több adatstruktúra alkalmazható, de szélesebb körben a relációs adatbázisokat használják. Ennek matematikai hátterét E. F. Codd fejlesztette ki az 1960-as évek végén. Egy relációs adatbázis olyan adattáblák gyűjteménye, melyek tetszőleges számú oszlopot és sort tartalmazhatnak. A táblák oszlopai névvel vannak ellátva, és általában minden táblának van egy, vagy összetett kulcs használata esetén több olyan oszlopa, mely egyértelműen azonosítja az adott tábla sorait. Ezt a speciális oszlopot szokás a tábla elsődleges kulcsának nevezni. A tábla soraira egyedként szokás hivatkozni, ahol az adott sorban szereplő értékek gyűjteménye jelenti az egyed tulajdonságait. Fontos, hogy a tábláink között különböző kapcsolatokat kell definiálnunk, melyekkel lehetőségünk van egy komplex adattábla logikailag kisebb egységekre való bontására. A tábláinkat összekapcsoló relációkat idegen kulcsok használatával szokás megvalósítani. Az idegen kulcsok mindig egy másik tábla elsődleges kulcsainak értékeit vehetik csak fel, ezáltal biztosítva a két tábla egyértelmű összekapcsolását.” [23]

A PostgreSQL egy robusztus, eleinte kicsit lassabb, de szinte mindenre kiterjedő adatbázisrendszer volt, mely „a Berkeley-i University of Californián 1986-ban elkezdett Postgres szoftverfejlesztésből származó Postgres95 továbbfejlesztett változata.” [24] A MySQL egy más fejlesztési irányvonalat követve fejlődött ki a Postgres mellett. Sikerének legfőbb oka, hogy megőrizte egyszerűségét és praktikusságát ezzel sikerült a mindennapi használatra alkalmas adatbázissá válnia. Nyílt forráskódú, ingyenes adatbázisrendszerről van szó, mely implementálja az SQL programnyelvet. A Unix és a

Linux különböző verziói mellett Windows rendszerekre is könnyen telepíthető, szemben a Postgres kimondottan nehézkes Windowsos telepítésével.

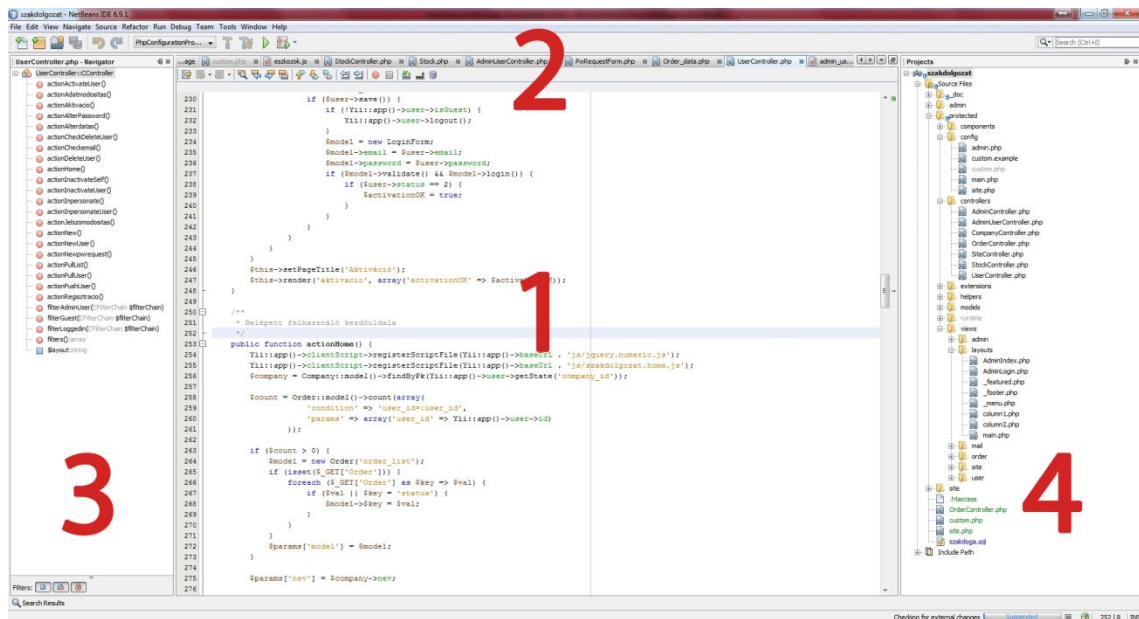
3.2. Fejlesztői eszközök

3.2.1. Netbeans

A programkódok megírásához használhatunk akármilyen szövegszerkesztőt, de jobban járunk, ha olyan programot veszünk igénybe, ami direkt szoftverfejlesztéshez lett kifejlesztve. Az ilyen programok sok segítséget tudnak nyújtani a fejlesztés közben, egyrészt jelzik, ha szintaktikai hibát vétünk, másrészt hiperhivatkozás-szerűen „ugrálhatunk” a kódban az egyes függvények, objektumok nevére kattintva. Ezeket a programokat integrált fejlesztői környezetnek vagy röviden IDE-nek (Integrated Development Environment) hívják. Ezek közül számos program ingyenesen letölthető és használható. Mivel az én alkalmazásomban háromféle programozási nyelvet is használok (PHP, HTML, Javascript), olyan programra volt szükségem, ami mindhárom nyelvet támogatja. A két legnépszerűbb IDE az Eclipse és a NetBeans. Objektíven itt is nehéz dönteni a kettő között, hiszen mindkét IDE nagyon sokat tud, sok kiegészítővel rendelkezik, gyakran frissítik, és nagy közösség áll mögöttük. Én személy szerint évek óta a NetBeans-t használtam PHP fejlesztésre, ezért most is e mellett döntöttem.

A hivatalos honlapról [25] ingyenesen le lehet tölteni, a szakdolgozat írásakor a 7.1.2-es verzió volt a legfrissebb. Letöltéskor kiválaszthatjuk, hogy milyen csomagokra van szükségünk. PHP fejlesztéshez elég a mindössze 47MB méretű telepítőt letöltenünk, ez tartalmazza a HTML és Javascript támogatást is.

Ahhoz, hogy ki tudjuk használni a NetBeans által nyújtott támogatást, a fejlesztéseinket projektekbe kell csoportosítanunk. Később, ha valamiért újra kell telepítenünk az operációs rendszert vagy a NetBeans-t, az elmentett projektadatokkal ott folytathatjuk a munkát, ahol előtte abbahagytuk. A NetBeans felépítése látható a 11. ábrán.



11. ábra
NetBeans IDE működés közben

A NetBeans IDE részei:

1. Szerkesztőfelület, itt írjuk a kódot. Ha szintaktikai hibát vétünk, a NetBeans figyelmeztet. Lehetőség van kódkiegészítésre, illetve automatikus kódgenerálásra is. Például, ha írunk egy osztályt, a NetBeans le tudja hozzá generálni a szükséges *getter*-t, *setter*-t, létrehozza a *konstruktor*-t, stb. Ha projektbe van foglalva az alkalmazás, akkor ki is tudja egészíteni a kódunkat. Elég, ha elkezdjük leírni egy függvény, osztály vagy metódus nevét, felkínálja a lehetőségeket.
2. Egyszerre több fájlt is megnyithatunk, az ábrán látható módon egymás mellé listázza őket a program, így gyorsan tudunk váltani a különböző kódok között. A listában az elemeket tetszőlegesen áthelyezhetjük.
3. A képen 3-ossal jelölt terület az ún. navigátor. A navigátor tetszőlegesen áthelyezhető a jobb oldalra is, illetve be is csukhatjuk, elég ha csak akkor látszik, amikor ténylegesen szükségünk van rá. Nagyobb osztályoknál lehet fontos szerepe, itt láthatjuk kilistázva az osztály metódusait, változóit.
4. A 4-essel jelölt terület szintén áthelyezhető, a képen a projekt fájllai láthatók rajta. Segít, hogy átlássuk a projektünk mappaszerkezetét, hamarabb megtaláljunk fájlokat.

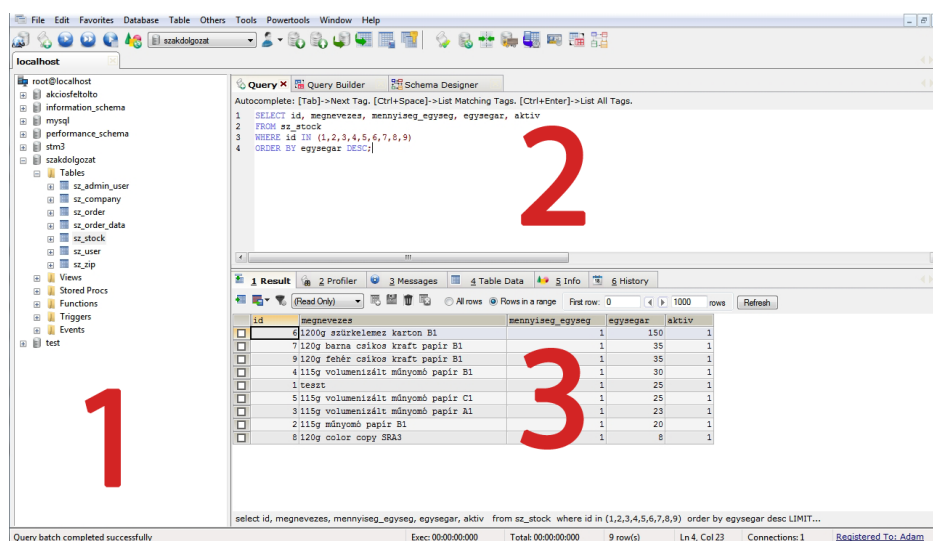
3.2.2. *SQLYog*

Az alkalmazásfejlesztés sikeréhez hozzátartozik az átgondolt, strukturált adatbázisterv készítése. Az adatbázisterv egy vizuális tervezési eszköz, melyen az adatbázis komponenseit láthatjuk: az adatbázistáblákat, azok kapcsolatait, a függéseket, elsődleges kulcsokat. Valamint szükséges ahhoz is egy eszköz, hogy a fejlesztés és a használat során nyomon tudjuk követni az adatbázis változásait.

Erre több ingyenes és fizetős megoldás is rendelkezésünkre áll, ilyen például a phpMyAdmin [26], azonban az adatbázisterv készítésre ez nem alkalmas. Mivel nekem adatbázisterv készítésre is szükségem volt, a *Webyog* cég *SqlYog* [27] nevű programjának 8-as, ingyenes verzióját használtam fel erre a célra (sajnos az újabb kiadásoknak már nincs ingyenes verziója, csak *trial* – azaz időkorlátos).

Főbb szolgáltatásai a programnak:

- MySQL menedzselése, úgy mint
 - adatbázisok és táblák létrehozása, törlése, módosítása, exportálása, importálása
 - kulcsok kezelése
 - SQL parancsok futtatása
- backup készítése (időzítetten is)
- bonyolult, összetett lekérdezések építése grafikus segítséggel (*query builder*)
- grafikus adatbázisterv készítése (*schema designer*)



12. ábra
SqlYog működés közben

Az SqlYog részei:

1. A webszerveren futó adatbázisok listája. Itt tudunk újat létrehozni, a jelenlegieket módosítani, kulcsokat kezelni, stb.
2. Szerkesztőfelület, ide írhatjuk az SQL parancsokat, melyeket a program képes kiegészíteni a táblák, mezők neveivel. A különböző parancsokat pontosvesszővel kell elválasztani, így le tudjuk futtatni egyszerre az egészt, vagy külön-külön is.
3. A kiválasztott adatbázisnak vagy táblának látjuk a részleteit, illetve az SQL lekérdezés eredményét.

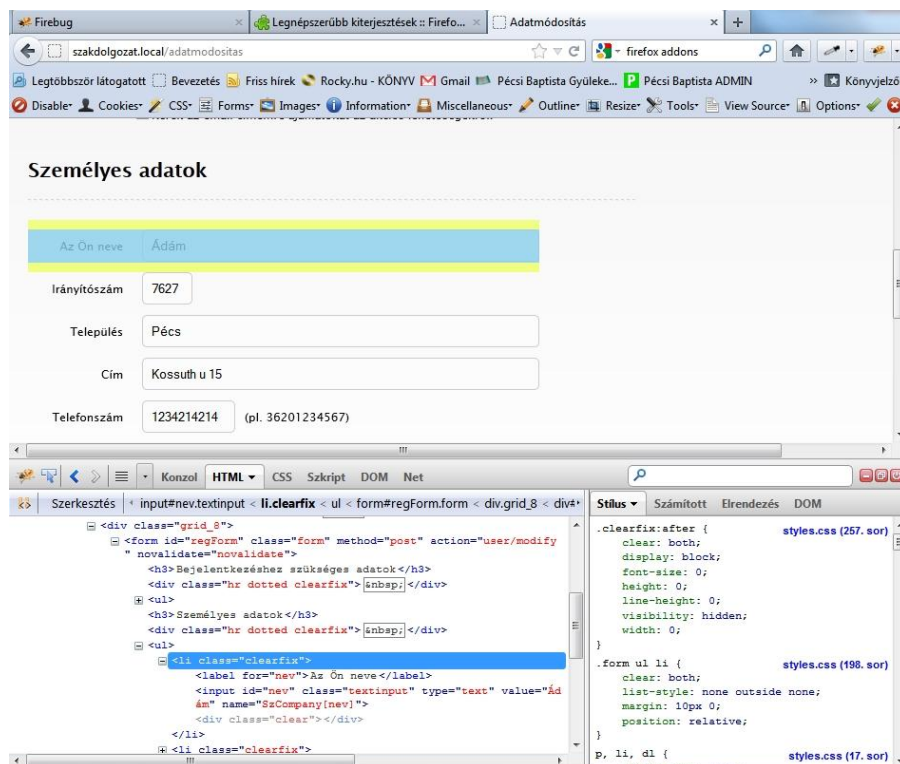
3.2.3. Firebug

A Firebug bármilyen jellegű webfejlesztéshez elengedhetetlen eszköz, valójában egy kiterjesztés a Mozilla Firefox-hoz. A mottója: *Web development evolved*, ami magyarul kb. így hangzik: a webfejlesztés kiterjesztése. Az első verziót 2006. decemberben adták ki, fő fejlesztője Joe Hewitt, korábbi Firefox fejlesztő. A Firefox Addons oldaláról eddig közel 47 millióan töltötték le! [28] Segítségével bármilyen weblapon módosíthatjuk, debuggolhatjuk és megtekinthetjük a CSS-t, HTML-t és Javascriptet, ami nagy segítség fejlesztés közben.

Számos kiegészítő érhető el hozzá, amivel még teljesebbé tehetjük, pl. Firecookie (cookie-k kezeléséhez), FirePHP (php debuggoláshoz), FlashFire (flash alkalmazás debuggoláshoz), stb.

Hasonló eszköz érhető el a Google Chrome-hoz (*Google Chrome Developer Tool*), az Internet Explorerhez (8-as verziótól: *Web Developer Tool*), de a szélesebb körű támogatottság és a sok kiegészítő miatt én a Firebug-ot használtam a fejlesztéshez.

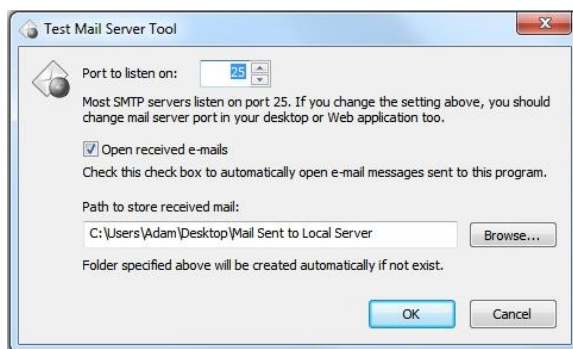
A 13. ábrán látható egy példa a Firebug alkalmazásának.



13. ábra
Firebug működés közben

3.2.4. Test Mail Server Tool

Mivel az alkalmazás a működése során többször is e-mailt kell, hogy küldjön, szükséges egy eszköz, amivel ezt tesztelni lehet a fejlesztés alatt is. A Test Mail Server Tool egy teljes értékű SMTP szerver-emuláló eszköz, ami alapból a localhost 25-ös portját figyeli (ez konfigurálható), és az itt kimenő e-maileket elmenti, vagy beállítástól függően egyből meg is jeleníti.



14. ábra
Test Mail Server Tool beállítása

3.3. Verziókövetés

A verziókövető rendszerek arra szolgálnak, hogy a programkódunk minden változatát elmentsék és segítsék a korábbi verziókra való visszatérést és a csoportmunkát. Láthatjuk ki melyik fájlt módosította, mikor, miért, a változtatások pedig könnyen összevethetőek egymással. A szakdolgozatom során egyedül csináltam az alkalmazást, mégis jó hasznát vettem a verziókövetésnek, mert így két gépen is (asztali és laptop) tudtam haladni a fejlesztéssel, attól függően, hogy épp hol voltam, valamint utólag is nyomon tudom követni, mikor mit csináltam.

Csoportmunka során pedig elengedhetetlen a verziókövetés, mert egy remek segítség abban, hogy ne kelljen egymásra várni, hanem egyszerre lehessen dolgozni különböző részeken, sőt, akár ugyan azon a fájlban is.

A verziókövetéshez szükséges egy szerver, ami a verziókezelésbe vont fájlokat tárolja. Első alkalommal feltöltünk egy induló változatot és utána minden egyes újbóli feltöltésnél a rendszer azt fogja eltárolni, hogy melyik fájlok változtak és miben. Ezzel a módszerrel biztosítja azt, hogy lépésenként láthassuk, hogy hogyan változtak a fájlok és bármelyik verzióra vissza tudjunk térni.

A verziókövető rendszerek általában kliens-szerver alapúak: a fejlesztők a szerveren tárolt ún. *repository*-ből jutnak hozzá a fájlokhoz (régi és új verziókhoz egyaránt; ez az *update* művelet), és a változtatásaikat is először a repository-ba küldik be (ez a *commit* művelet), a többi fejlesztő már a repository-ból jut hozzá a változtatásokhoz.

Két nagy csoportra bonthatjuk a verziókövetőket: elosztott és központosított.

A központosított verziókövetők a régebbiek, ma már kevésbé népszerűek, itt ugyanis minden változás csak egy helyen, a szerveren van tárolva. Ilyen például a *Subversion*.

Az elosztott verziókövető azt jelenti, hogy bár van egy központosított szerver, minden felhasználónál megtalálható a központi szerveren található repository tulajdonképpeni teljes archívuma (*backup*-ja). Egy üzemzavar, vagy meghibásodás esetén ezek bármelyike visszatérhet a szerverre, hogy lecserélje a központi szerveren levő példányt.

Az elosztott verziókövető rendszer további jellemzője, hogy nem igényel állandó, megbízható internet-kapcsolatot a központi szerverrel, és emiatt sokkal kevesebbszer

fordul a szerverhez, mint nem elosztott társai (tehát, mivel a legtöbb művelet a helyi repository-ra hat, jóval kevesebb hálózati forgalmat generál, és az átlagos válaszideje is kisebb). A két legnépszerűbb és legelterjedtebb elosztott verziókövető a *Git* és a *Mercurial*.

Az alábbiakban a legfontosabb fogalmakat és műveleteket ismertetem: [29]

- **Repository** (*tároló*): egy távoli szerver, ezen találhatóak a fájlok
- **Revision** (*változat*): a szerveren található fájlok egy adott változata
- **Commit** (*elkönyvelés*): a helyi változtatások szerverre feltöltése
- **HEAD revision** (*fő változat*): a szerveren található legújabb állapot
- **BASE revision** (*alapváltozat*): a munkakönyvtárban található legfrissebb állapot
- **Differences** (*eltérésmutatás*): két különböző állapot összevetésének naplója
- **Merge** (*összefűzés*): két különböző változat egybefűzése (előfordul, hogy nem sikerül tökéletesen, ilyenkor összetűzés (*conflict*) alakul ki)
- **Conflict** (*összetűzés*): tökéletlen összefűzés (*merge*) során kialakuló helyzet
- **Resolve** (*feloldás*): az összetűzés (*conflict*) megoldása (ilyenkor a munkatárs önyvelés előtt átnézi a konfliktusba került módosításokat és végrehajt egy sikeres összefűzést)
- **Checkout**: a távoli szerver egy adott állapotának lemásolása a gépeden található munkakönyvtárba
- **Working copy** (*munkakönyvtár*): a fájlokat tartalmazó mappa a számítógépeken, ennek tartalma kerül könyvelésre a szerverre.
- **Trunk** (*törzs*): a fő fejlesztési ág
- **Branch** (*ág*): a fő ággal és egyéb ágakkal párhuzamosan fejlesztett ágak
- **Tag** (*megjelölt változat*): egy lezárt fejlesztési ág vagy kiadás mappája

A fenti ismertetésből már látszik, hogy mindenképp érdemes elosztott rendszert használni, még akkor is, ha csak egyedül fejlesztünk valamit. Én személy szerint használtam már mind a három fentebb említett verziókövetőt, és a szubjektív véleményem alapján a Mercurial használata mellett döntöttem. A Subversion azért esett ki a választásból, mert nem elosztott, a Git és a Mercurial között az döntött, hogy a Mercurial-hoz elérhető egy remek grafikus kezelőfelület, a *Tortoise HG*, és ennek a kezelése számomra szimpatikusabb és egyszerűbbnek tűnik, mint a Git kezelése.

Ahhoz, hogy használni tudjuk a verziókövetőt, szükségünk van egy központi szerverre, ahol létrehozhatjuk a repository-t (tároló). Három népszerű, ingyenes megoldást ismerek, de ezeken kívül valószínűleg vannak még, így itt is nehéz a választás. A *Google code* [30] és a *GitHUB* [31] nagy népszerűségnek örvend, én mégis a *Bitbucket* [32] mellett döntöttem, ugyanis az előbbi kettővel ellentétben itt ingyen csinálhatunk nem publikus projektet is. A Bitbucket támogatja a Git-et és a Mercurial-t, tehát ebből a szempontból is megfelelő számomra. Regisztráció után létre tudunk hozni egy repository-t, majd a saját gépünkre le kell *clone*-oznunk (vagyis egy lokális másolatot csinálunk). Innentől kezdve a Tortoise HG segítségével végezhetjük a verziókövetést.

Ennek segítségével több lokális ágat (branch-et) tarthatunk fenn, amelyek teljesen függetlenek egymástól. Ezen fejlesztési vonalak létrehozása, egyesítése és törlése csupán másodpercekbe kerül.

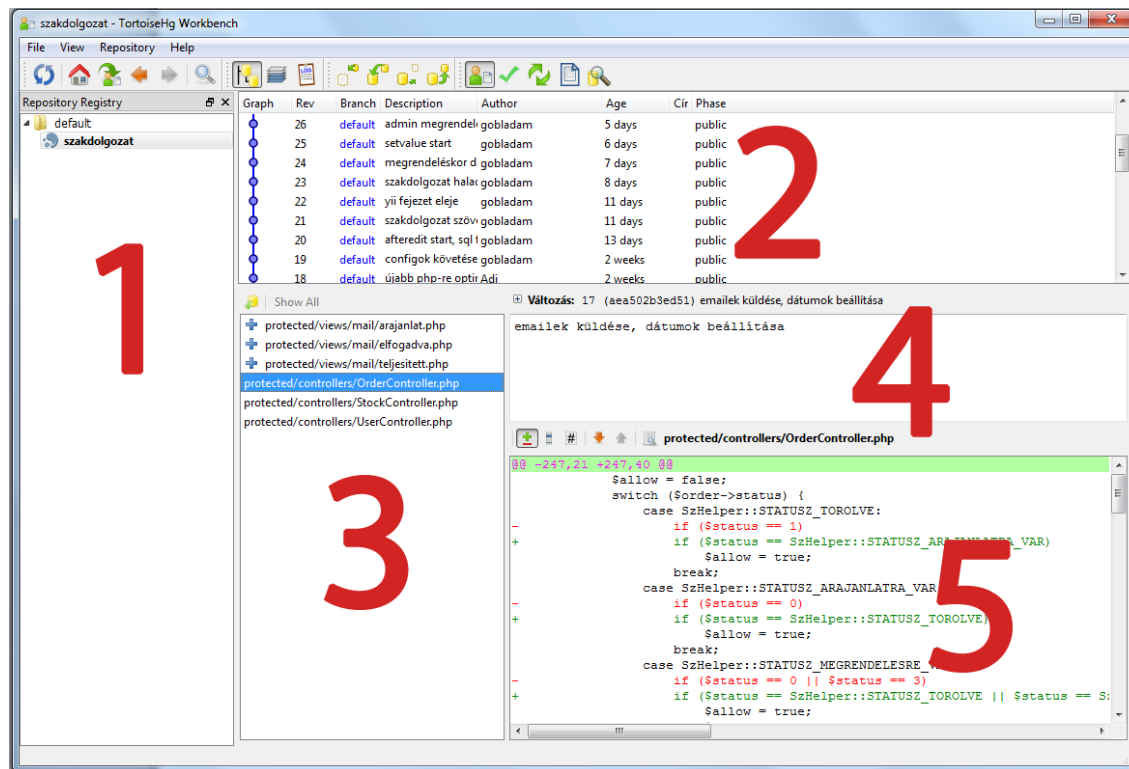
Ez azt jelenti, hogy olyan dolgokat tehetünk, mint például:

- Létrehozhatunk egy új ágat, hogy kipróbálhassunk egy új ötletet, néhány változtatás (commit) után visszaválthatunk oda, ahonnan nyitottuk az új ágat, összefésülhetjük az azóta fejlesztett dolgokkal, majd visszaváltunk a kísérletező helyre, és beolvastjuk.
- Fenntarthatunk egy ágat, amiben mindig csak az éles rendszerbe kerülő dolgok vannak (ezt szokás *master*-ágnak hívni), egy másikat, amibe a tesztelésre kerülő munkákat olvasztjuk be, és több kisebbet a mindennapi feladatokra.
- Minden újabb feladatra, amelyen dolgozunk, új ágat hozhatunk létre, hogy gond nélkül váltogathassunk közöttük. Ezek után mindegyiket törölhetjük, amikor az adott dolog beolvasztásra kerül a főágban.
- Létrehozhatunk egy új ágat kísérletezésre; ha esetleg nem jött be a kísérletezés, egyszerűen kitörölhetjük, lemondva az abban végzett munkáról. Ezt az egészet senki más nem látja (akkor sem, ha közben más ágakat feltöltöttünk).
- Amikor egy távoli tárolóba (repositroy-ba) feltöltjük a változásokat, akkor nem kell minden ágat felrakni, elég csak azt, amin épp dolgoztunk.

A projekteken belül általában vannak olyan fájlok, amiket nem szükséges követni. Ilyenek például a lokális konfigurációs fájlok, amik minden fejlesztőnél mások (pl. adatbázis-kapcsolat, elérési utak, stb.), vagy az ideiglenes létrehozott fájlok. Egy

.HGIGNORE nevű fájlban beállíthatjuk, hogy ezek a fájlok ne legyenek követve, így mielőtt commit-olnánk, nem is látszódnak, hogy ezek változtak.

Az én szakdolgozatom esetében ilyen nem követett állomány a *custom.php*, ami az adatbázis-elérés konfigurációja és a cache-elés beállításait tartalmazza. Helyette egy *custom.example* fájl van követve, amiben mindig jelzem a változásokat.



15. ábra
Tortoise HG működés közben

A Tortoise HG működését a 15. ábra szemlélteti. A felület részei a következők:

1. A követett repository-k listája.
2. A revision-ök listája. Itt láthatjuk a különböző branch-eket is.
3. Ha kiválasztunk egy commit-ot, akkor a 3-as ponttal jelzett területen nézhetjük meg a fájlokat, amiket tartalmaz az adott commit. Ha commit-olni akarunk, akkor pedig a commitolandó fájlok listáját látjuk itt.
4. Ha kiválasztottunk egy commit-ot, akkor a 4-es ponttal jelzett területen a leírása látszik. Ha commitolni szeretnénk, akkor ide írhatjuk meg a commit leírását.
5. A 3-as pont-ban kiválasztott fájl változásait mutatja. Zölddel az új sorokat, pirossal a törölteket. (a színezés konfigurálható)

A bitbucket egy másik szolgáltatása az ún. hibajegykezelés. Az esetek többségében nem egyedül fejlesztünk egy alkalmazást, hanem egy csoport tagjaként veszünk részt a projektben. A csoportnak van egy vezetője, az ő feladata kiosztani a feladatokat a fejlesztők között. Erre a dologra több szoftveres megoldás is született, ilyenek a hibajegy-kezelők is. A magyar terminológia hibajegyként emlegeti az angol „ticket” szót, de az egyes hibajegyek nem feltétlenül tényleges hibákat, bugokat jelentenek, hanem az egyes feladatokat tűzzük ki bennük.

Eddig kétféle hibajegy-kezelőt volt alkalmam kipróbálni, az egyik a fizetős *Fogbugz*, a másik az ingyenes *Redmine*. Mindkettő alkalmas projektek menedzselésére, vannak különböző jogosultsági szintek, lehet definiálni szabályokat, kapcsolatokat az egyes feladatok között, valamint mérni is lehet a feladatokra fordított időt.

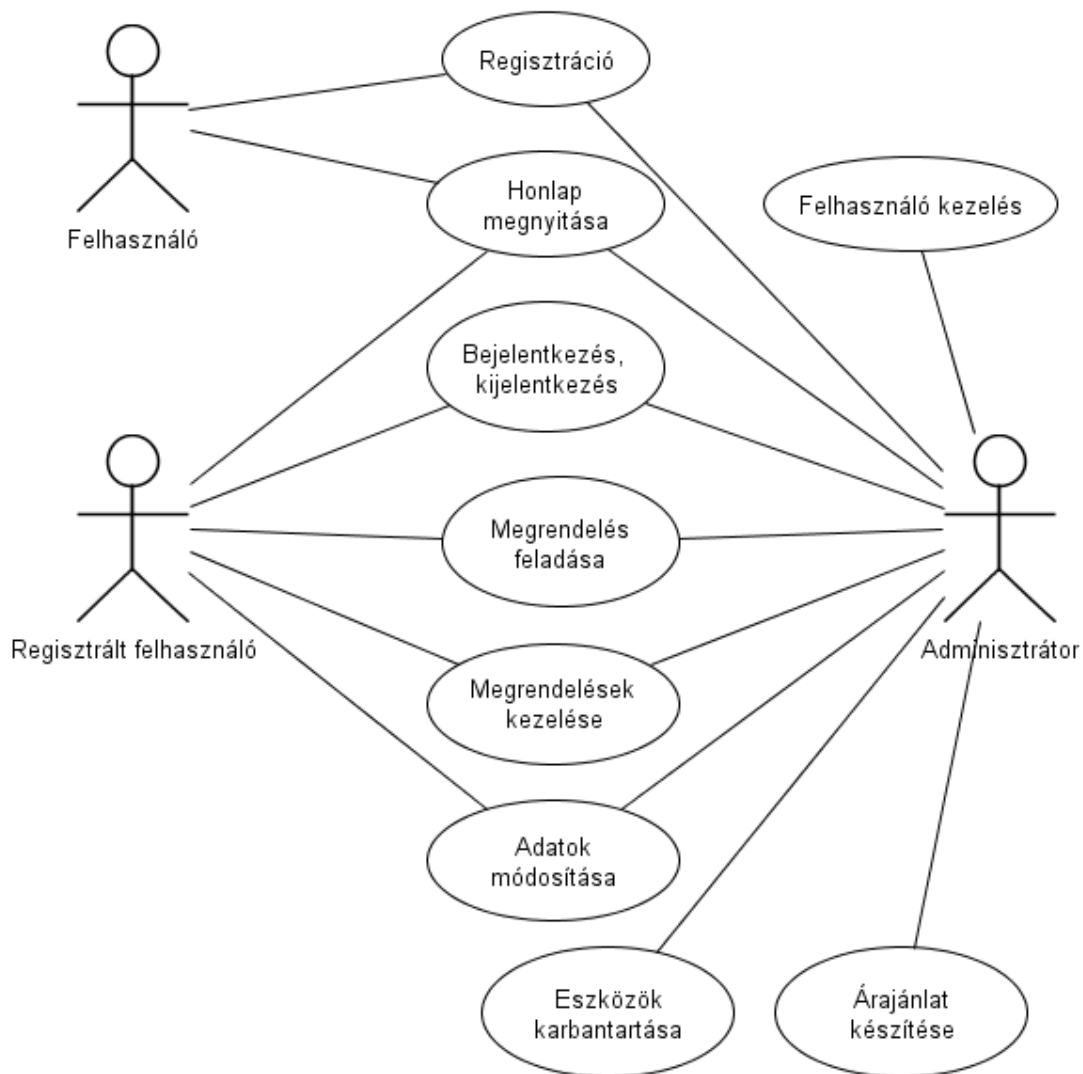
A bitbucket beépített hibajegy-kezelő rendszere az angol *issue* szóval jellemzi az egyes feladatokat és 5 különböző fejlesztőt vehetünk be a projektekbe ingyenesen. A feladatokat hozzárendelhetjük a fejlesztőkhöz, és minden feladatnak megadhatjuk a következő adatokat:

- Feladat rövid elnevezése
- Leírás, fájlt is csatolhatunk hozzá
- Kihez tartozik
- Milyen jellegű feladat (hiba, fejlesztés, stb.)
- Prioritás

A fejlesztők e-mail-ben kapnak értesítést, ha egy feladatot hozzájuk rendeltünk.

3.4. Használati eset diagram

Mielőtt nekiállunk elkészíteni bármilyen alkalmazást, célszerű végiggondolni, hogy kik fogják használni, milyen szerepkörök lesznek, és milyen funkciókat érhetnek el. Az oldal felépítését és működését legjobban a használati eset diagram segítségével lehet modellezni, leírni. A webalkalmazás használati eseti diagramja a 16. ábrán látható. A webalkalmazásban a diagramon feltüntetett funkciók lesznek megvalósítva.



16. ábra
Az alkalmazás használati eseti diagramja

3.5. Az adatmodell tervezése

A rendszer alapjául egy 7 táblából álló SQL adatbázis szolgál. A továbbiakban ezeknek a tábláknak a felépítését, funkcionalitását ismertetem. A teljes adatmodell a táblák között levő kapcsolatokkal a 3.5.5-os fejezetben található.

Mivel az alkalmazás karakterkódolása *UTF-8*, ezért az adatbázis táblák is ezt a karakterkészletet használják.

3.5.1. Felhasználók tárolása

A használati eseti diagramból már látható, hogy az alkalmazást háromféle felhasználó használhatja. Ezek közül kettőt, az adminisztrátorokat és a regisztrált felhasználókat tároljuk adatbázisban.

Adminisztrátorok tárolása

Az adminisztrátor felhasználók adatait az *sz_admin_user* táblában tároljuk. Az alkalmazás működésének feltétele, hogy minimum egy aktív adminisztrátor legyen. Az alkalmazás telepítésekor egy adminisztrátor van, ő az ún. *superadmin*, neki mindenre van jogosultsága. Adminisztrátor felhasználót csak olyan adminisztrátor tud regisztrálni, akinek van jogosultsága az admin felhasználók kezelésére. A rendszer a működés során biztosítja, hogy minimum egy aktív adminisztrátor legyen, mégpedig úgy, hogy az adminisztrátor nem törölheti ki saját magát a felületen keresztül, ha nincs több aktív adminisztrátor rajta kívül.

SZ_ADMIN_USER

| | | |
|----------|--------------|--------|
| id | integer(11) | PK, NN |
| name | varchar(100) | NN |
| username | varchar(50) | NN |
| password | char(32) | NN |
| email | varchar(100) | NN |
| phone | varchar(20) | NN |
| mobile | varchar(20) | NN |
| created | datetime | NN |
| active | tinyint(1) | NN |
| roles | text | NN |

1. táblázat
Adminisztrátor tábla

Az 1. táblázatban láthatjuk a tábla mezőit. Az alapvető személyes adatokon túl (*name*, *email*, *phone*, *mobile*) a többi mező a hitelesítést, belépéshez szükséges adatokat tárolja. Minden adminisztrátornak van egy felhasználóneve (*username*) és egy hozzá tartozó jelszó (*password*), ezzel a két adattal tudja magát hitelesíteni belépéskor. A jelszavakat MD5-ös kódolás után tároljuk, ami egy 128 bites egyirányú kódolási algoritmus, amit gyakorlatilag lehetetlen visszafejteni. Mivel „az MD5 kódolás bármilyen adatból –

függetlenül a méretétől, vagy a típusától – egy 32 karakter hosszú hexadecimális hasht eredményez” [33], elég, ha egy 32 karakter hosszú *char*-ként tároljuk. A *created* mezőben elmentjük, hogy az adott adminisztrátor mikor lett létrehozva. Az *active* mezőben tároljuk, hogy az adminisztrátor használhatja-e a rendszert. Ha egy felhasználóról tudjuk, hogy hosszabb ideig nem fogja használni az alkalmazást, de később valószínű újra fogja, akkor nem muszáj kitörölnünk, elég ha inaktíváljuk. Az *active* mező vagy 0 vagy 1 értéket vehet fel, ezért elegendő, ha *tinyint*-ként tároljuk 1 karakteren. A *roles* mezőben találhatók a jogosultságok, szerializált PHP tömbként. Háromféle jogosultságot adhatunk az adminoknak:

- admin felhasználó: ezzel a jogosultsági szinttel tudja kezelni a többi felhasználót, adminokat és regisztrált felhasználókat egyaránt: létrehozhat, törölhet, aktiválhat
- beszállító: olyan külső munkatárs, aki a felhasználható eszközöket tudja menedzselni: létrehozhat, módosíthat, törölhet, aktiválhat.
- szerkesztő: ezzel a jogosultsággal láthatja a megrendeléseket, tud árajánlatot adni, módosítani, törölni.

Felhasználók tárolása

| SZ_USER | | | SZ_COMPANY | | |
|------------|--------------|--------|------------|--------------|--------|
| id | integer(11) | PK, NN | id | integer(11) | PK, NN |
| password | char(32) | NN | nev | varchar(255) | NN |
| company_id | integer(11) | FK, NN | irszam | varchar(4) | NN |
| status | mediumint(9) | NN | varos | varchar(64) | NN |
| check_code | varchar(127) | | cim | varchar(64) | NN |
| email | varchar(127) | NN | sz_nev | varchar(255) | NN |
| created | datetime | NN | sz_irszam | varchar(4) | NN |
| last_login | datetime | | sz_varos | varchar(64) | NN |
| edm | tinyint(1) | | sz_cim | varchar(64) | NN |
| | | | tel | varchar(16) | NN |

2. és 3. táblázat

Felhasználók tárolása

Az oldal felhasználóit, akik a megrendeléseket tudják feladni, két táblában tároljuk. A hitelesítéshez szükséges adatokat és a személyes, valamint számlázási adatokat külön

tábla kezeli. Ez egyrészt a továbbfejlesztési lehetőségek miatt van így, illetve az eredeti alkalmazásban is így volt megoldva annak érdekében, hogy a későbbiek során egy számlázási címhez több felhasználó is tartozhasson.

A hitelesítéshez szükséges adatokat az *sz_user* táblában tároljuk. A felhasználó az e-mail címe és a hozzá tartozó jelszó megadásával tud belépni az oldalra. A *status* mezőben tároljuk a felhasználó státuszát. Regisztráció után szükséges, hogy aktiválja magát, csak ezután tudja használni az oldalt. Eltároljuk a regisztrációja dátumát (*created*) valamint minden belépéskor felülírjuk a *last_login* mező tartalmát. Az *edm* mező szintén a bővíthetőség miatt van itt: ha a felhasználó engedélyezi a hírleveleket az oldal üzemeltetői részéről, akkor azt ebben a mezőben tároljuk (*electronic direct marketing*). A *company_id* idegen kulcs, az *sz_company* tábla valamelyik elsődleges kulcsa lehet az értéke. Ezzel kapcsoljuk össze a hitelesítő adatokat és a személyes adatokat.

Az *sz_company* táblában tároljuk a felhasználó személyes és számlázási adatait, ezek a következők: teljes név, irányítószám, város, cím és telefonszám.

3.5.2 Városok és irányítószámok

A regisztrációs űrlapok kitöltésének megkönnyítése miatt az *sz_zip* táblában tároljuk a magyarországi városok irányítószámát és a hozzájuk tartozó városok nevét, valamint a megyéket (a megyéket egyenlőre nem használja az alkalmazás, ez egy bővítési lehetőség). Amikor a felhasználó kitölti az irányítószám mezőt, a program magától kitölti a város mezőt. Ezt a táblát az eredeti alkalmazásból emeltem át ide, egyébként az adatbázis tartalma a Magyar Posta Zrt. nyilvántartásából származik. Csv-formátumban ingyen le lehet tölteni a honlapjukról, majd egy PHP feldolgozó szkripttel migráltam az adatbázisba.

SZ_ZIP

| id | integer(11) | PK, NN |
|-----------|--------------|--------|
| zip | integer(4) | NN |
| city | varchar(210) | NN |
| county_id | integer(11) | NN |

4. táblázat
Irányítószámok és városok

3.5.3 Felhasználható eszközök

Hogy a különböző megrendelésekre árajánlatot tudjon adni az alkalmazást használó cég, le kell tárolnunk a rendelkezésre álló eszközöket is. A különböző fajta papírokat és nyomdaipari technológiákat az eredeti alkalmazásból vettem át. Az *sz_stock* táblában tároljuk ezeket.

SZ_STOCK

| | | |
|------------------|--------------|--------|
| id | integer(11) | PK, NN |
| megnevezes | varchar(255) | NN |
| mennyiseg_egyseg | tinyint(1) | NN |
| egysegar | integer(11) | NN |
| aktiv | tinyint(1) | NN |

5. táblázat
Rendelkezésre álló eszközök

Az *aktiv* mezőben tároljuk, hogy jelenleg elérhető-e az adott eszköz (van-e raktáron). Csak olyan eszközöket tudnak felhasználni az adminisztrátorok árajánlat készítésekor, amik aktívak. A *mennyiseg_egyseg* mezőben a mennyiségi egység azonosítóját tároljuk, magukat a mennyiségi egységeket a *main.php* konfigurációs fájlban tároljuk, tömb formájában. Bővítési lehetőség, hogy ezeket is táblában tároljuk, és menedzselni lehessen.

3.5.4 Megrendelések nyilvántartása

A felhasználók megrendeléseit két táblában tároljuk. Az *sz_order* táblában a megrendelés adatait tároljuk, a megrendelés teljesítéséhez szükséges eszközök listáját és darabszámukat pedig az *sz_order_data* táblában.

A *user_id* mező idegen kulcs, ezzel kapcsoljuk össze a felhasználók táblával a megrendelést. A *status* mezőben a megrendelés státuszát tároljuk. Maguk a státuszok konstansként vannak definiálva egy ún. helper osztályban. Jelenleg 5 státusz van.

```
const STATUSZ_TOROLVE = 0;  
  
const STATUSZ_ARAJANLATRA_VAR = 1;  
  
const STATUSZ_MEGRENDELESRE_VAR = 2;  
  
const STATUSZ_MEGRENDELVE = 3;  
  
const STATUSZ_TELJESITETT = 4;
```

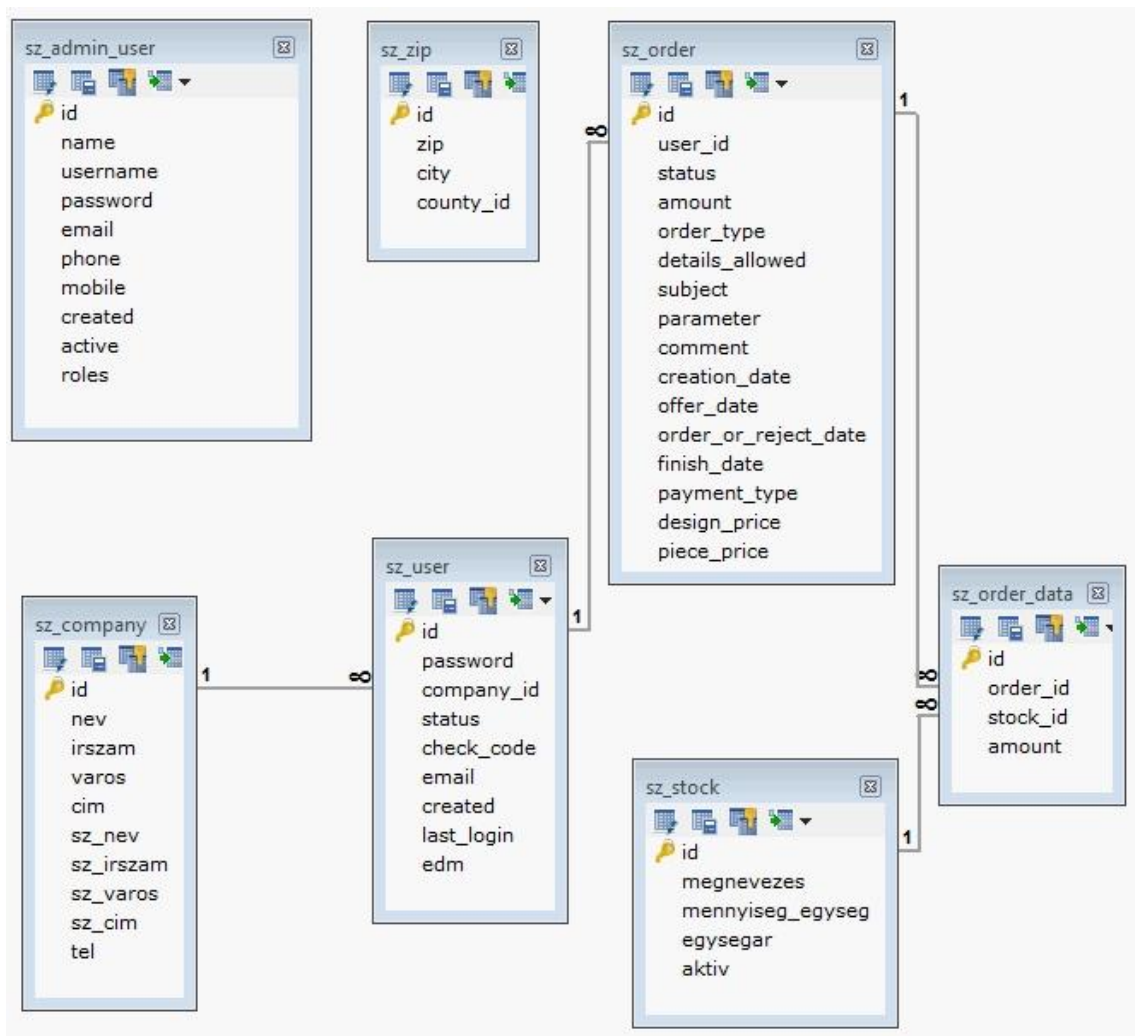
Az *amount* mezőben a darabszámot tároljuk, az *order_type*-ban a megrendelés típusát (pl. szórólap, újság, stb.). A típusokat is a *helper* osztályban definiáltam, bővítési lehetőség, hogy menedzselhetők legyenek. A megrendelés rövid megnevezését a *subject* mezőben tároljuk, míg a hosszabb leírását a *parameter* mezőben. Az adminisztrátor tetszőleges kommentet fűzhet az egyes megrendelésekhez, ezt a *comment* mező tárolja. A megrendelés feladásának dátumát a *creation_date* mezőbe mentjük le, a többi dátum mezőt az árajánlat alakulásától függően töltjük ki: *offer_date* az árajánlat adás dátuma, *order_or_reject_date* a megrendelés vagy visszautasítás dátuma, a *finish_date* pedig a megrendelés teljesítésének dátuma. A felhasználandó eszközök árából és darabszámából számoljuk ki, hogy egy darab termék előállításához mennyibe kerül, ezt a *piece_price* mezőben tároljuk, míg a tervezési költséget a *design_price* mező tartalmazza.

| SZ_ORDER | | | SZ_ORDER_DATA | | |
|----------------------|--------------|--------|---------------|-------------|--------|
| id | integer(11) | PK, NN | id | integer(11) | PK, NN |
| user_id | integer(11) | FK, NN | order_id | integer(11) | NN |
| status | tinyint(2) | NN | stock_id | integer(11) | NN |
| amount | integer(11) | NN | amount | integer(11) | NN |
| order_type | integer(11) | NN | | | |
| details_allowed | tinyint(1) | NN | | | |
| subject | varchar(255) | NN | | | |
| parameter | text | NN | | | |
| comment | text | | | | |
| creation_date | date | | | | |
| offer_date | date | | | | |
| order_or_reject_date | date | | | | |
| finish_date | date | | | | |
| payment_type | tinyint(1) | | | | |
| design_price | integer | | | | |
| piece_price | integer(11) | | | | |

6. és 7. táblázat

Megrendelések tárolása

3.5.5 A teljes adatmodell



17. ábra

Az alkalmazás által használt adatmodell

4. A WEBALKALMAZÁS ELKÉSZÍTÉSE

4.1. Kódgenerálás

Most, hogy nagyjából ismertettem a Yii Framework tulajdonságait, megterveztem az adatmodellt, nincs más hátra, mint megírni a kódokat. A Yii ebben is segítségünkre van a Gii nevű kiegészítőjével. A Gii a Yii Framework kódgeneráló alkalmazása, az 1.1.2-es verziótól elérhető. A 2.4-es fejezetben bemutatam, hogy tudunk létrehozni egy alkalmazás vázat (*skeleton application*), most ismét ezt vesszük elő, mint kiindulási állapot.

A Gii elindításához be kell állítanunk az adatbázis-kapcsolatot, és még néhány dolgot, ezeket a `<webszerver>/alkalmazasneve/protected/main.php`-ben tehetjük meg.

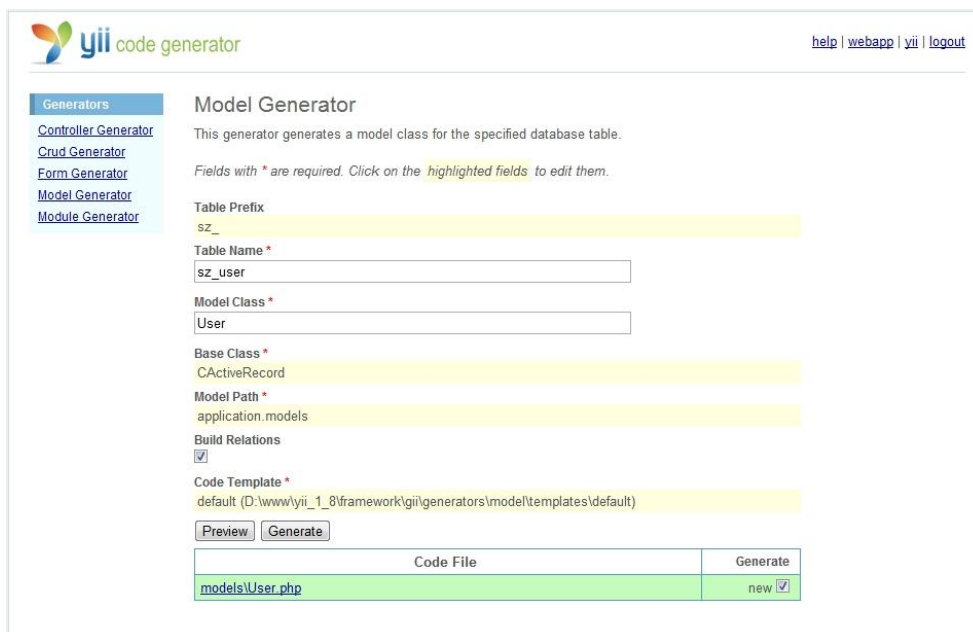
```
return array(
    ...
    'import'=>array(
        'application.models.*',
        'application.components.*',
    ),
    'modules'=>array(
        'gii'=>array(
            'class'=>'system.gii.GiiModule',
            'password'=>'jelszot_kell_ide_irni',
        ),
    ),
    'db'=>array(
        'class' => 'CDbConnection',
        'emulatePrepare' => true,
        'charset' => 'utf8',
        'connectionString' =>
'mysql:host=localhost;dbname=szakdolgozat',
        'username' => 'root',
        'password' => 'adatbazis_jelszava',
    ),
    ...
);
```

Ha mindent jól állítottunk be, akkor a következő URL-en érjük el a Gii-t: `http://<webszerver>/alkalmazasneve/index.php?r=gii`. A konfigurációs fájlban megadott jelszó beírása után öt menüpont közül választhatunk:

- Controller generator
- Crud generator
- Form generator
- Model generator
- Modul generator

Gyakorlatilag egy komplett alkalmazást ki tud nekünk generálni a Gii az adatbázisunk alapján. Ahogy korábban már írtam róla, a Yii rugalmas, így megtehetjük, hogy csak bizonyos kódrészeket generáltatunk ki vele, és azokat is átírhatjuk. Az alkalmazásomhoz elegendő, ha a modelleket legeneráltatom, a többi kódot (megjelenítést, formokat) magam írom meg.

A *Model Generator* menüpontba lépve egy űrlap fogad bennünket, ahol meg kell adnunk annak a táblának a nevét, aminek a modelljét ki akarjuk generáltatni, valamint hogy van-e prefix (nálam ez a konvencióknál már ismertetett `sz_`). A Gii modell-generáló formot a 18. ábra szemlélteti.



The screenshot shows the 'yii code generator' interface. On the left, a sidebar lists 'Generators' with links to 'Controller Generator', 'Crud Generator', 'Form Generator', 'Model Generator', and 'Module Generator'. The 'Model Generator' is selected. The main area is titled 'Model Generator' and contains the following fields and options:

- Table Prefix:** sz_ (highlighted in yellow)
- Table Name *:** sz_user
- Model Class *:** User
- Base Class *:** CActiveRecord (highlighted in yellow)
- Model Path *:** application.models (highlighted in yellow)
- Build Relations:** ☒
- Code Template *:** default (D:\www\yii_1_8\framework\gii\generators\model\templates\default) (highlighted in yellow)

Below the form are 'Preview' and 'Generate' buttons. At the bottom, there is a table with two columns: 'Code File' and 'Generate'.

| Code File | Generate |
|-----------------|---|
| models\User.php | new <input checked="" type="checkbox"/> |

18. ábra
Gii kódgeneráló – a képen a User modell generálása látható

A továbbiakban bemutatom egy-egy példán keresztül, hogy hogy néz ki egy *modell*, egy *controller* és egy *view*, később csak néhány helyen térek ki a konkrét kódra, miközben bemutatom az elkészült alkalmazást képekkel illusztrálva (5. fejezet).

4.1.1. A user modell bemutatása

A user modell a külső (tehát nem adminisztrátor) felhasználókat reprezentálja. Ahogy a 2.3.1-es fejezetben írtam, a Yii Frameworkben kétféle modell van: a user modell aktív rekord, tehát egy tábla adatait reprezentálja, változói a tábla mezői.

A modell osztályokban kell definiálni az adott modell kapcsolatait, és szabályait. Ez a user modell esetében így néz ki:

```
/**
 * @return array kapcsolatok
 */
public function relations()
{
    return array(
        'company' => array(self::BELONGS_TO, 'Company',
            'company_id'),
        'order' => array(self::HAS_MANY, 'Order', 'user_id')
    );
}

/**
 * @return array attribútumokra vonatkozó szabályok
 */
public function rules()
{
    return array(
        array('password, company_id, status, email, created',
            'required'),
        array('email', 'email'),
        array('password', 'required'),
        array('password', 'length', 'min' => 1),
        array('password', 'length', 'max' => 255),
        array('check_code, email', 'length', 'max' => 127)
    );
}
```


4.1.2. A user controller bemutatása

A user controller a külső felhasználókra vonatkozó kérélmeket kezeli, a *CController* osztály leszármazottja. A Yii Framework controllerei általában két dologból állnak: filterek és action-ök. A Filter az a kódrészlet, amely az action futtatása előtt és/vagy után fut. Itt tudunk definiálni ellenőrzéseket, például hogy bizonyos action-ök csak akkor futhatnak le, ha a felhasználó be van jelentkezve.

Nézzünk egy példát a filterre:

```
/**
 * Szűrők hozzárendelése
 * @return array
 */
public function filters() {
    return array(
        ...
        'guest + regisztracio',
        ...
    );
}
```

A filterek olyan sorrendben futnak le, ahogy fel vannak véve a tömbbe. A fenti példának az a lényege, hogy a regisztrációs űrlap csak nem belépett (tehát vendég – *guest*) felhasználók esetében fog megjelenni. Először a program ellenőrzi a controller-ben definiált *filterGuest()* metódust, és ha nincs belépve a felhasználó, meghívja az *actionRegisztracio()* metódust, egyébként átirányít a kezdőoldalra.

Nézzük, hogy is néz ki a *filterGuest()* és az *actionRegisztracio()*:

```
/**
 * csak vendégek érhetik el az oldalt
 * @param CFilterChain $filterChain
 */
public function filterGuest($filterChain) {
    if (!Yii::app()->user->isGuest) {
        $this->redirect(array('site/index'));
    } else {
        $filterChain->run();
    }
}
```

```

/**
 * Regisztrációs űrlap
 */
public function actionRegisztracio() {
    Yii::app()->clientScript->registerScriptFile(Yii::app()-
>baseUrl . 'js/szakdolgozat.regisztracio.js');
    ...
    $this->setPageTitle('Göbl Ádám - szakdolgozat regisztrációs
oldal');
    $this->render('regisztracio');
}

```

Ha vendégként használjuk az oldalt, akkor eljutunk az *actionRegisztracio()* metódusba. Itt csak annyit csinálunk, hogy a *registerScriptFile()* segítségével behívjuk az egyedi Javascripteket, beállítjuk az oldal címét, majd rendereljük az oldalt.

4.1.3. A user view-k bemutatása

Ahogy a 2.3.1-es fejezetben már ismertettem, a view-k szolgálnak a megjelenítésre. A view neve megegyezik a PHP script fájl nevével, melyet a *protected/views/ControllerID* mappában kell elhelyezni. A *UserController* megjelenítő fájljai a fentiek szerint a *protected/views/user* mappában vannak.

Az előző példánál maradva, az *actionRegisztracio()* a *regisztracio* nevű view-t rendereli, aminek az elnevezése megegyezik a view nevével, tehát *regisztracio.php*.

A renderelés során a kiválasztott view-t egy layout-ba ágyazza a Yii. A controller elején definiálhatjuk az alapértelmezett layout-ot is, de ezt bármelyik action-ben felülírhatjuk. A szakdolgozat alkalmazásában végig egy layout-ot használtam, így minden controller első sorai közt megtalálható az alábbi kód:

```

/**
 * Alapértelmezett layout
 * @var string
 */
public $layout = 'application.views.layouts.column1';

```

A *column1.php*-ban mindössze ennyi kód van:

```

$this->beginContent('application.views.layouts.main');
    echo $content;
$this->endContent();

```

A *\$content* helyére teszi be kiválasztott view-t, de előtte behívja a *main* nevű layout-ot. A *main.php* tartalmaz minden olyan megjelenítéshez szükséges adatot, ami minden oldalbetöltéshez szükséges. Itt van definiálva a *doctype*, itt vannak behívva az alap CSS-ek és Javascript-ek, a menü, és az alsó sáv.

4.2. ExtJS Javascript Library

Ahogy a célkitűzések között már írtam, az adminisztrátori felület megjelenését az ExtJS Javascript Library biztosítja. Az ExtJS szintén egy keretrendszer, azonban a teljes bemutatása kitenne egy újabb szakdolgozatot, ezért most csak röviden ismertetem.

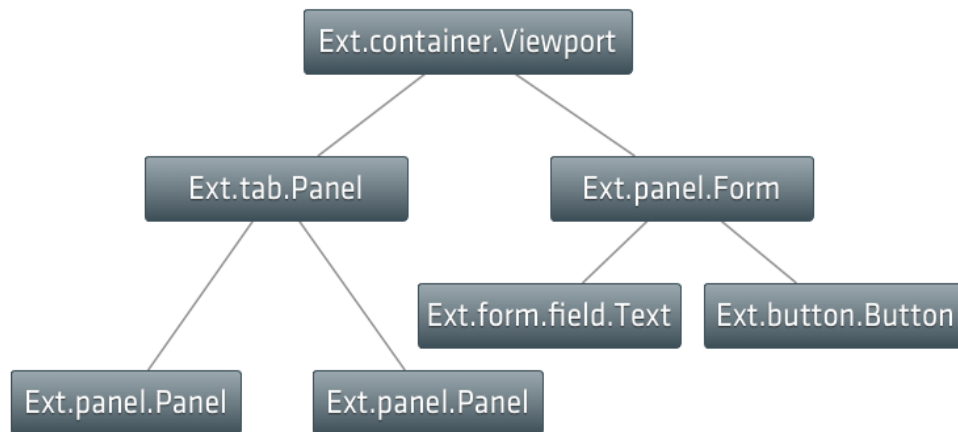
Tehát, az ExtJS egy kiforrott Javascript GUI keretrendszer, ami nagyon sok jól megvalósított kliens oldali widgetet tartalmaz. Nagy előnye, hogy böngésző független (IE6+, FF1.5+, Safari 3+, Chrome 3+, Opera 9+). Kitűnő támogatással rendelkezik: jó API dokumentációja van sok példával, aktív, segítőkész közösség használja, rendszeresen tartanak képzéseket, konferenciákat. A neve az *Extension* angol szóból származik, ami magyarul kb. így hangzik: hozzáépítés, kibővítés, kiterjesztés. Az első verziót 2007-ben adták ki, ma már a 4.1-es verziónál tartanak. Általában adminisztrátor felületekhez használják.

Az ExtJS is megvalósítja az MVC modellt. Az ExtJS ún. *store* objektumokat használ a modell megvalósításához. A *view*-k tetszőleges komponensek (pl. grid táblázat), a *controllerek* a modellek inicializálásáért, *view*-k rendereléséért és egyéb alkalmazáslogikákért felelősek. A *controllerek* eseményekre figyelnek, melyek bekövetkezésekor valamilyen tevékenységet folytatnak. [34]

Egy ExtJS alkalmazás grafikus felülete egy vagy több widget-ből épül fel, amiket komponensnek nevezünk. Minden komponens az *Ext.Component* osztályból származik, ez az osztály biztosítja az automatizált életciklus menedzselését, beleértve:

- inicializálás
- renderelés
- átméretezés
- pozicionálás
- megsemmisítés

Az ExtJs rengeteg komponenst tartalmaz, továbbá lehetőségünk van saját komponensek elkészítésére. A konténer (*container*) olyan komponens, ami más komponenseket képes tárolni. A konténer felelős a gyermekei életciklusának kezeléséért. Egy alkalmazás általában legalább egy konténert tartalmaz: a komponenshierarchia tetején lévő *Viewport*, amely az összes többi konténert és komponenst tartalmaz.

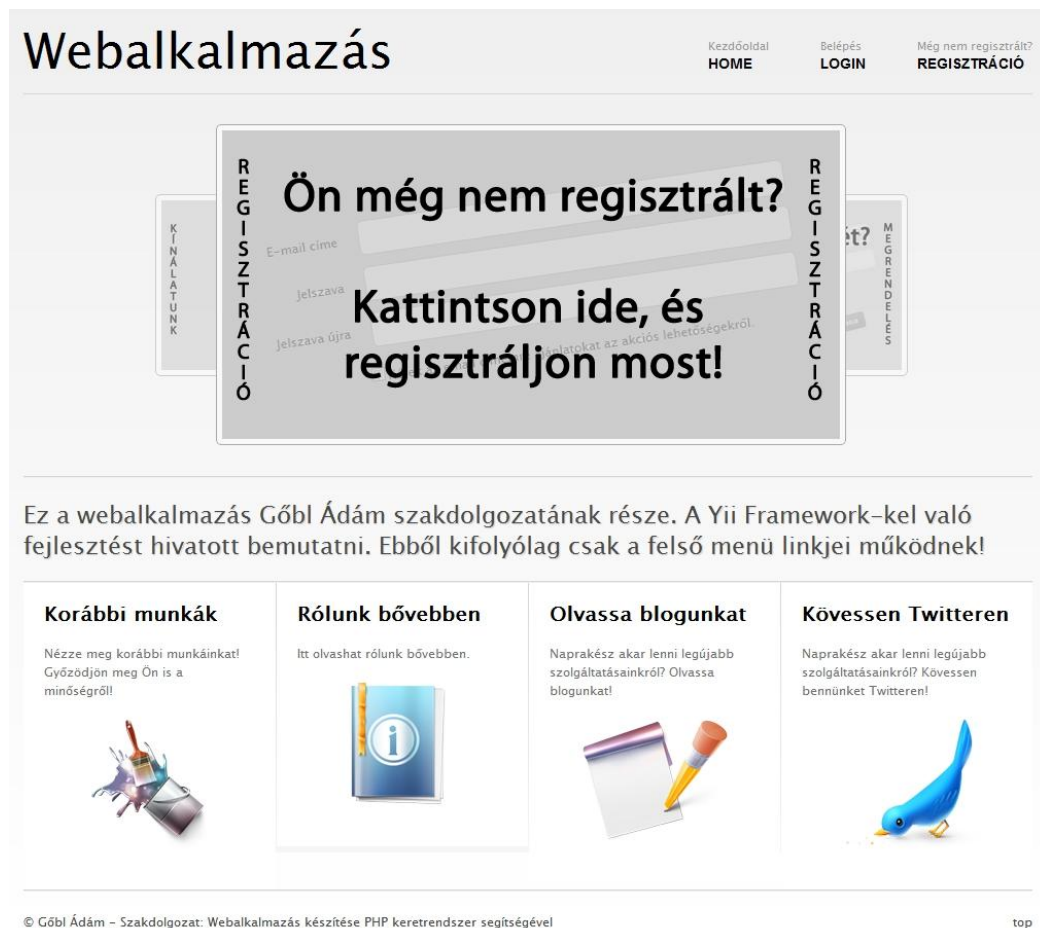


19. ábra
ExtJS komponenshierarchia [35]

5. A WEBALKALMAZÁS MŰKÖDÉS KÖZBEN

5.1. Az internetes megrendelés felülete

Az alkalmazás egyik fő modulja az internetes megrendelés felülete, ami gyakorlatilag a cég publikus honlapja is lehetne egyben. A szakdolgozathoz csak a szükséges részeket valósítottam meg, így a 20. ábrán látható kezdőlapon a linkek nagy része nem működik, csak a felső sorban található menüpontok. A mai web-dizájn irányzatoknak megfelelően igyekeztem kialakítani az oldalt, így egységes betűtípusokat és méreteket alkalmaztam a hasonló funkciójú és jelentéstartalmú részekhez, valamint a figyelmeztetések, pl. hibás űrlapkitöltés esetén ún. *prompt*-ban [36], vagyis felugró ablakok formájában jelennek meg.



20. ábra
Nyitólap

A nyitólapot a *SiteController*-ben található *actionIndex()* szolgálja ki és az *index* nevű *view* jeleníti meg.

A felhasználók a nyitólapról a regisztrációs űrlapra tudnak navigálni, vagy ha már az oldal regisztrált felhasználói, be tudnak lépni, ill. elfelejtett jelszó esetén új jelszót tudnak igényelni. A felhasználókat belépéskor az e-mail címükkel és a hozzá tartozó jelszóval hitelesíti az alkalmazás.

Új jelszó igénylése esetén a felhasználónak meg kell adnia azt az e-mail címét, amivel korábban regisztrált az oldalra. Ha valós címet adott meg, a rendszer törli az e-mail címhez tartozó jelszavát, és véletlenszerűen generál egy újat, majd e-mailben elküldi a felhasználónak. Azért van szükség az új, véletlenszerűen generált jelszóra, mert a rendszer titkosítva tárolja a jelszavakat, így az eredetileg megadott jelszót nem tudja közölni a felhasználóval.

Ha valaki rendelést szeretne feladni, ahhoz muszáj regisztrálnia az oldalon. A nyitólapról elérhető a regisztrációs oldal, melynek egy lehetséges kimenetét láthatjuk a 21. ábrán.

A honlap azon részeit, ahol a felhasználók közvetlenül kommunikálhatnak az alkalmazással (pl. űrlapok esetén), védetté kell tenni. Ellenőrizni kell, hogy csak megfelelő adatok kerülhessen be az adatbázisba. Az alkalmazás az ellenőrzéseket minden űrlap esetén két szinten végzi: kliens és szerver oldalon egyaránt.

A kliens oldali validációt egy Javascript kiegészítő segítségével végzem az oldalon található összes űrlap esetében. [37] Ha valamelyik mezőbe formailag nem megfelelő adatot írunk be, vagy üresen hagyunk egy kötelező mezőt, akkor az adott mező alatt magyarul tájékoztatja a felhasználót, hogy milyen adatokat lehet beírni a mezőbe. Addig nem engedi elküldeni az űrlapot, amíg nincs minden helyesen kitöltve.

A regisztrációs űrlap három szekcióra van osztva:

- Belépéshez szükséges adatok (e-mail cím, jelszó)
- Személyes adatok (név, irányítószám, település, cím, telefonszám)
- Számlázási adatok (név, irányítószám, település, cím)

Az e-mail cím esetén kétféle ellenőrzés is van, egyrészt hogy szabványos e-mail címet adott-e meg a felhasználó, másrészt hogy nem regisztrált már korábban ezzel az e-mail címmel. Mivel az e-mail cím a hitelesítéshez kell, egy e-mail címet csak egyszer lehet regisztrálni.

Szeretné feladni megrendelését? **Regisztráljon most!**

Bejelentkezéshez szükséges adatok

E-mail címe

Jelszava

Jelszava újra

☒ Kérek az email címemre ajánlatokat az akciók lehetőségeiről.

Személyes adatok

Az Ön neve

Irányítószám

Település

Cím

Telefonszám (pl. 36201234567)

Számlázási adatok

☐ Ugyan az, mint a személyes adatoknál megadottak

Név
A mező kitöltése kötelező

Irányítószám
A mező kitöltése kötelező

Település
A mező kitöltése kötelező

Cím
A mező kitöltése kötelező

Tovább

21. ábra
Regisztrációs űrlap

Ha a személyes adatok és a számlázási adatok megegyeznek, az oldal segítséget nyújt, hogy ne kelljen kétszer beírni ugyan azokat az adatokat. Ehhez be kell pipálni a számlázási adatok résznél található „Ugyan az, mint a személyes adatoknál megadottak” gombot, erre az alkalmazás kitölti a számlázási adatok mezőit a személyes adatoknál megadottakkal.

A regisztrációs űrlapot a *UserController*-ben található *actionRegisztracio()* metódus szolgálja ki és a *regisztracio* nevű *view* jeleníti meg.

Sikeres regisztráció esetén a felhasználó e-mailben kap egy linket, aminek segítségével hitelesítheti (aktiválhatja) a regisztrációt. Csak sikeres aktiváció után tud belépni az oldalra.

A belépett felhasználók megváltoztathatják a regisztrációkor megadott adataikat. Ehhez be kell lépniük az oldalra, és az „Adataim” menüponton keresztül érhetik el az adatmódosító űrlapot.

Az adatmódosító űrlapot a *UserController*-ben található *actionAdatmodositas()* metódus szolgálja ki és ugyan az a *view* jeleníti meg, mint a regisztrációs űrlapot (kód-újrahasznosítás végett). A felhasználó itt megváltoztatja az adatait, akár az e-mail címét is. Ez esetben a felhasználónak aktiválnia kell az új e-mail címét, valamint az alkalmazás ki is lépteti.

The screenshot shows the 'Webalkalmazás' application interface. At the top, there are navigation links: 'Kezdőoldal HOME', 'Adataim módosítása ADATAIM', 'Új rendelés RENDELÉS', and 'Kilépés LOGOUT'. Below the navigation bar, a greeting 'Üdvözljük, Ádám!' is displayed. A search bar indicates 'A keresési feltételeknek 8 rendelés felel meg (látszik 1 - 5)'. Below the search bar is a table with 6 columns: 'Azonosító', 'Típus', 'Rövid megnevezés', 'Mennyiség', 'Státusz', and 'Megtekintés'. The table contains 6 rows of data. At the bottom of the table, there are pagination controls: '< Előző', '1', '2', 'Következő >'. The footer of the page shows '© Göbl Ádám - Szakdolgozat: Webalkalmazás készítése PHP keretrendszer segítségével' and a 'top' link.

| Azonosító | Típus | Rövid megnevezés | Mennyiség | Státusz | Megtekintés |
|-----------|-----------|------------------------|-----------|-------------------|-------------|
| 1 | folyóirat | átírom adminban | 15 db | Teljesített | Részletek |
| 2 | dosszié | ezi s | 1 db | Teljesített | Részletek |
| 4 | képeslap | Karácsonyi képeslap | 40 db | Teljesítés alatt | Részletek |
| 6 | karszalag | Karszalag konfira | 150 db | Megrendelésre vár | Részletek |
| 8 | képeslap | Árvíztűrő tükörfúrógép | 15 db | Megrendelésre vár | Részletek |

22. ábra

Belépés után a korábbi rendelések listáját látjuk

A 22. ábrán látjuk a belépést követő nyitólapot és az elérhető menüpontokat. A nyitólapot a *UserController*-ben található *actionHome()* metódus szolgálja ki és a *home* nevű *view* jeleníti meg.

A nyitólapon van egy rendezhető, szűrhető, lapozható táblázat, amiben a korábbi rendeléseiről kap információkat a felhasználó. Fel van tüntetve a rendelés egyedi azonosítója, a típusa, rövid megnevezése, mennyisége és státusza. Minden sor végén található egy link, amivel az adott rendelés részletei tekinthetők meg.

A táblázat a Yii Framework *CGridView* nevű komponense, aminek segítségével egyszerű volt megoldani, hogy az oldal újratöltése nélkül lehessen rendezni, szűrni, lapozni a táblázatban.

Új rendelés feladásához a menüpontok közül a „Rendelés”-t kell választani, ekkor a rendelés űrlapjához navigál az oldal, amit a 23. ábra szemléltet.

Webalkalmazás

Kezdőoldal HOME Adataim módosítása ADATAIM Új rendelés RENDELÉS Kilépés LOGOUT

Rendelés feladása

Típus:

Megnevezés:

Darabszám:

Részletek:

Fizetés:

Megrendelés feladása

© Göbl Ádám – Szakdolgozat: Webalkalmazás készítése PHP keretrendszer segítségével top

23. ábra
Megrendelés űrlap

A következő adatokat kell megadni megrendelés feladásához: az előállítani kívánt termék típusa (pl. füzet, katalógus, naptár stb.), egy rövid megnevezés, darabszám, az elképzelések részletezése, valamint hogy milyen módon történik a fizetés.

Ha az adatok validálása során nem lépett fel hiba, a rendszer elmenti a megrendelést, a státusza „Árajánlatadásra vár” lesz. Az adminisztrátor(ok) e-mailt kapnak róla, hogy árajánlatot kell adniuk (a *custom.php* konfigurációs fájlban az *adminEmail* paraméterben lehet beállítani, hogy ki kapjon e-mailértesítést ilyenkor).

A megrendelés űrlapot az *OrderController*-ben található *actionRendeles()* metódus szolgálja ki és a *rendeles* nevű *view* jeleníti meg. Amíg nem érkezett árajánlat, a felhasználó módosíthatja az igényeit: a kezdőlapon található táblázatban kikeresheti a rendelést, a „részletek” linkre kattintva a rendelés részleteinek oldalára jut, aminek egy lehetséges kimenetét a 24. ábra szemlélteti.

Webalkalmazás

Kezdőoldal
HOME

Adataim módosítása
ADATAIM

Új rendelés
RENDELÉS

Kilépés
LOGOUT

Megrendelés neve

Itt tart az Ön rendelése: (50 %)

Megrendelésre vár

MEGRENDEL

TÖRÖL

Árajánlat részletei

Árajánlat: 2012. 05. 11.

Tervezési díj: 5000 HUF

Darabár: 1500 HUF

Összesen: 27500 HUF

Felhasználható eszközök

4 db: Boríték

5 db: boríték beigazítás – 1 szín

3 db: boríték beigazítás – 2 szín

5 db: boríték nyomás – 1 szín (1000)

Rendelés részletei

Feladás dátuma: 2012. 04. 23.

Típus: folyóirat

Megnevezés: Megrendelés neve

Darabszám: 15

Részletek: paraméter módosítva

második sor

harmadik sor

negyedik

adminban

Fizetés: Készpénz

© Göbl Ádám – Szakdolgozat: Webalkalmazás készítése PHP keretrendszer segítségével

top

24. ábra
Árajánlat után a megrendelés részletei

57

A megrendelés részletei oldalt az *OrderController*-ben található *actionDetails()* metódus szolgálja ki és a *details* nevű *view* jeleníti meg. Az oldal tetején grafikusán van ábrázolva, hogy hol tart a rendelés folyamata. A jobb oldalon a rendelés státusza látható, alatta a státusztól függően gombok vannak.

Az alábbi státuszokkal rendelkezhet egy rendelés:

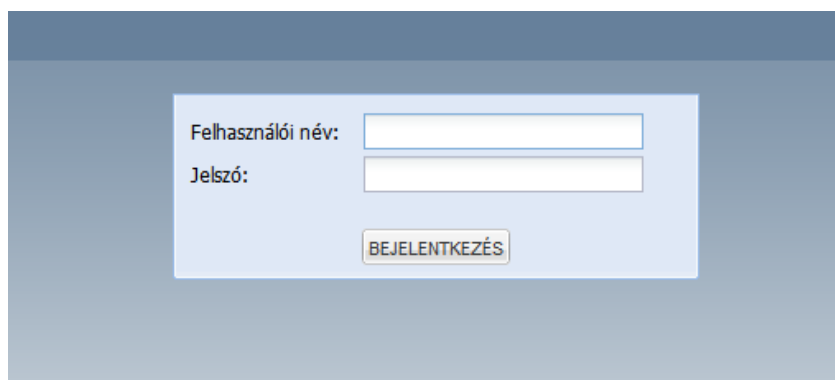
- Törölt
- Árajánlatra vár
- Megrendelésre vár
- Megrendelve
- Teljesített

Törölt státuszt azért vehet fel egy rendelés, mert az eredeti alkalmazás alapján az adatbázisból nem törölünk ki egy megrendelést sem a későbbi statisztikák miatt.

Törölni azokat a megrendeléseket lehet, amik árajánlatra várnak, vagy már kapott rájuk a felhasználó árajánlatot, de nem fogadja el.

5.2. Az adminisztrációs modul

Az adminisztrációs modult a korábban ismertetett ExtJS jeleníti meg. Az admin modul megjelenítéséről két *view* gondoskodik, az egyik a *login* nevű, ami a bejelentkező űrlapot jeleníti meg, a másik az *index* nevű, itt jelenik meg a kezdőoldal, a további menüpontok dinamikus töltődnek be. A 25. ábrán látható a bejelentkező űrlap.



25. ábra

Az adminisztrációs modul bejelentkező űrlapja

Sikeres bejelentkezés után jogosultságtól függően jelennek meg a menüpontok. Az alábbi jogosultsági szintek vannak: (részletesebben a 3.5.1-es fejezetben volt róluk szó)

- admin felhasználó
- szerkesztő
- beszállító

Az admin felhasználó mind a négy menüpontot eléri (Admin felhasználók, Felhasználók, Megrendelések, Eszközök). A szerkesztő csak a „Megrendelések” menüpontot, a beszállító csak az „Eszközök” menüpontot tudja használni.

Az adminisztrátori modul felépítése felhasználótól függetlenül egyforma:

- bal oldali sávban találjuk az elérhető menüpontokat
- jobb felső sarokban van a „Kilépés” gomb
- jobb oldalon az adott menüpontból elérhető elemek dinamikus listája található, ami szűrhető, rendezhető, lapozható. A lista jobb felső sarkában van lehetőség új elemet létrehozni. A táblázat alján állíthatjuk be, hogy hány találatot jelenítsen meg, itt tudjuk frissíteni a táblázatot, valamint a jobb alsó sarokban kapunk információt, hogy összesen hány eleme van a táblázatnak.

Admin felhasználóként a belépést követően az „Admin felhasználók” menüpontba kerülünk, amit a 26. ábra szemléltet. A táblázatban az adminisztrátorok listája látható, halványoszürke színnel az inaktív felhasználók (ők nem is tudnak belépni).

Bejelentkezve: Superadmin Kilépés

Menü << Admin felhasználók

Admin felhasználók

ID/név/f. név/e-mail Keresés Új admin felhasználó

| ID | Név | Felhasználói név | E-mail | Telefon | Mobil |
|----|------------|------------------|----------------|---------|-------------|
| 13 | Nyomdász | nyomdasz | nyomd@asz.hu | 123456 | 123456 |
| 1 | Superadmin | superadmin | super@admin.hu | 123456 | 06305086366 |
| 14 | Szerkesztő | szerkeszto | szerk@esz.to | 123456 | 123456 |

Oldal 1 a 1-ből/ből 20 Admin felhasználók: 1 - 3/3

26. ábra
Adminisztrátorok listája

Ha valamelyik admin felhasználóra duplán kattintunk, vagy megnyomjuk az „Új admin felhasználó” gombot, akkor a 27. ábrán szemléltetett űrlap nyílik meg. Itt tudjuk beállítani az adott felhasználó adatait, ki tudjuk törölni, inaktíválhatjuk, beállíthatjuk a jogosultságait, valamint megváltoztathatjuk a jelszavát.

Nyomdász

Név:

Felhasználói név:

E-mail:

Telefon:

Mobil telefon:

Aktív: ☒

Szerepkörök

Admin felhasználó: ☐

Szerkesztő: ☐

Beszállító: ☒

Jelszó változtatás

Új Jelszó:

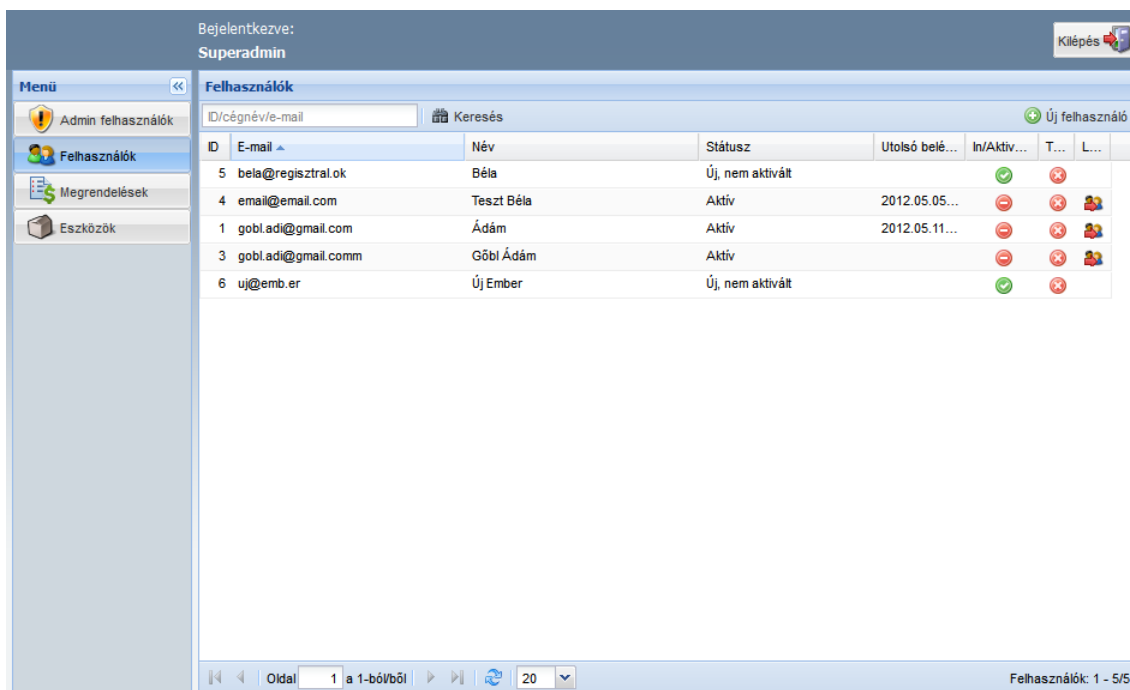
Még egyszer:

27. ábra

Adminisztrátor adatait állíthatjuk ebben a formban

Az internetes megrendelés felületén keresztül regisztrált felhasználókat a „Felhasználók” menüpontban kezelhetjük, amennyiben „Admin felhasználó” jogosultsággal használjuk a modult. A felhasználók listáját a 28. ábrán láthatjuk.

A táblázat minden elemének utolsó három oszlopában gombokat látunk. Az „In/Aktiválás” nevű oszlopban a piros ikonnal tudjuk inaktíválni az adott felhasználót, és a zöld ikonnal aktiválni. Mellette, a „Törlés” nevű oszlopban a piros ikonnal tudjuk törölni a felhasználót. Ez az egyetlen pontja az alkalmazásnak, ahol tényleges törlést csinálhatunk. Az eredeti alkalmazásban az volt a kérés, hogy ha itt törölünk egy felhasználót, akkor a hozzátartozó rendelések is törlődjenek. Ha nem akarjuk, hogy elvessenek a korábbi rendeléseik, érdemes inkább inaktíválni a felhasználót. Az utolsó, „Legyek ő” nevű oszlopban található ikonra kattintva a rendszer beléptet a megrendelő oldalra az adott felhasználó nevében.



28. ábra
Felhasználók listája

A felhasználóknak három státusza lehet:

- Új, nem aktivált – ha már regisztrált, de még nem aktiválta az e-mail címét
- Aktív – használhatja az oldalt
- Inaktív – nem tud belépni

Adminisztrátorként az e-mail címe aktiválását is elvégezhetjük a felhasználó helyett, ekkor e-mailben tájékoztatja őt a rendszer.

Ha valamelyik felhasználóra duplán kattintunk, a felnyíló űrlapon megváltoztathatjuk az e-mail címét és a jelszavát. Ha adminként változtatjuk meg az e-mail címét, akkor nem kell újra aktiválni a regisztrációt.

Ha a táblázat jobb felső sarkában található „Új felhasználó” gombra kattintunk, akkor a 29. ábrán látható űrlap nyílik fel, itt hozhatunk létre új felhasználót. Ugyan azok a validálások érvényesek itt is, mint amikor a felhasználó saját magát regisztrálja.

Ki kell tölteni a bejelentkezéshez szükséges adatait, a személyes- és számlázási adatokat. Sikeres létrehozás után a rendszer e-mailt küld a felhasználónak.

Új felhasználó

Bejelentkezéshez szükséges adatok

E-mail:

Jelszó:

☐ Hírlevél engedélyezve

Személyes adatok

Név:

Ir. szám, Település:

Cím:

Telefonszám:

☐ Ez a számlázási cím is egyúttal

Számlázási adatok

Név:

Ir.szám, Település:

Cím:

29. ábra
Új felhasználó létrehozása

Bejelentkezve: Superadmin Kilépés

Menü << **Megrendelések**

Admin felhasználók Felhasználók **Megrendelések** Eszközök

Megnevezés Keresés

| ID | Megnevezés | Megrendelő | Státusz | Darabszám | Típus | Feladás dátuma | Tervezési díj | Darabár | Összték |
|----|-------------------------------|-------------|-------------------|-----------|------------------|----------------|---------------|-----------|------------|
| 1 | Általános adminban | Ádám | Teljesített | 15 db | foliórat | 2012.03.02 | 15 Ft | 5 Ft | 90 Ft |
| 2 | ezt s | Ádám | Teljesített | 1 db | dosszié | 2012.03.11 | 15000 Ft | 50 Ft | 15050 Ft |
| 4 | Karácsonyi képeslap | Ádám | Megrendelve | 40 db | képeslap | 2012.03.31 | még nincs | még nincs | még nincs |
| 6 | Karszalag konfra | Ádám | Megrendelésre vár | 150 db | karszalag | 2012.04.20 | 5000 Ft | 40 Ft | 11000 Ft |
| 8 | Árvízűtő tükrűtűrógép | Ádám | Megrendelésre vár | 15 db | képeslap | 2012.04.21 | 1000 Ft | 40 Ft | 1600 Ft |
| 9 | Árvízűtő tükrűtűrógép | Ádám | Megrendelésre vár | 15 db | dosszié | 2012.04.21 | 1000 Ft | 10 Ft | 1150 Ft |
| 10 | Ez a meg | Ádám | Megrendelésre vár | 15 db | képeslap | 2012.04.21 | 50000 Ft | 40000 Ft | 650000 Ft |
| 11 | Megrendelés neve | Ádám | Megrendelésre vár | 15 db | foliórat | 2012.04.23 | 5000 Ft | 1500 Ft | 27500 Ft |
| 12 | Új led tv használati útmutató | Tesztt Béla | Teljesített | 500 db | használati út... | 2012.05.05 | 15000 Ft | 2500 Ft | 1265000 Ft |

Oldal: 1 a 1-ből/ből 20 Megrendelések: 1 - 9/9

30. ábra
Megrendelések listája

A felhasználók által feladott megrendeléseket a „Megrendelések” menüpontban lehet kezelni. A 30. ábrán látható a megrendelések táblázata.

A törölt illetve teljesített státuszú megrendelések halványszürkével jelennek meg, az árajánlatra váró és megrendelt státuszú rendelések félkövérrel.

Ha egy megrendelésre duplán kattintunk, a 31. ábrán látható űrlap nyílik meg. Az űrlapon két „fül” (*tab*) van, a „Rendelés adatai” fülön a rendelés részletes adatait tekinthetjük meg és módosíthatjuk (státusztól függően), a „Rendeléshez felhasználható eszközök” fülön – amit a 32. ábra szemléltet – az árajánlat teljesítéséhez szükséges eszközöket és darabszámukat adhatjuk meg.

A „saját komment” mezőbe lehet írni tetszőleges megjegyzést a megrendelésről, ezt a megrendelő nem fogja látni. Ha bepipáljuk az „Eszközöket láthatja-e” mezőt, akkor a felhasználó a megrendelése részleteinél tételesen fogja látni, hogy milyen eszközöket és azokból hány darabot használunk fel a rendelése teljesítéséhez.

Megrendelés - Megrendelés neve

Rendelés adatai | Rendeléshez felhasználható eszközök

Megnevezés:

Típus:

Mennyiség: db

Eszközöket láthatja-e: ☒

Részletek:

paraméter módosítva
második sor
harmadik sor
negyedik
arminhan

Saját komment:

Dátumok

Feladás dátuma: 2012-04-23

Ajánlat dátuma: 2012-05-11

Elfogadás vagy visszautasítás dátuma:

Árak

Tervezési díj: Ft

31. ábra
Megrendelés részletei űrlap

A szükséges eszközök összeállítását a „Rendeléshez felhasználható eszközök” fülre kattintva tehetjük meg. Ekkor két táblázatot látunk: a bal oldaliban az aktív státuszú eszközök vannak felsorolva. Azokat az eszközöket, amiket fel kívánunk használni a rendeléshez, át kell húzni a jobb oldali táblázatba. Miután áthúztuk, a darabszámmra kattintva tudjuk megadni, hogy hány darabra van szükség. Közben a „Rendelés adatai”

fülön két mező dinamikusan számolódik: az „Anyagköltség egy darabra” és az „Anyagköltség összes” mezők értékei segítenek abban, hogy az árajánlat összeállítója meg tudja állapítani a „darabár”-at, és a „tervezési költség”-et.

Ha kész van az árajánlat, az „Árajánlat mentése” gombbal tudjuk elmenteni. Ekkor a megrendelőt e-mailben tájékoztatja az alkalmazás, hogy árajánlata érkezett.

| Megnevezés | Kiszerezés | Egységár |
|------------------------------------|------------|----------|
| teszt | ív | 25 ft |
| 115g műnyomó papír B1 | ív | 20 ft |
| 115g volumenizált műnyomó papír A1 | ív | 23 ft |
| 115g volumenizált műnyomó papír B1 | ív | 30 ft |
| 115g volumenizált műnyomó papír C1 | ív | 25 ft |
| 1200g szűrkelemes karton B1 | ív | 150 ft |
| 120g barna csikos kraft papír B1 | ív | 35 ft |
| 120g color copy SRA3 | ív | 8 ft |
| 120g fehér csikos kraft papír B1 | ív | 35 ft |
| 120g ofszet papír A1 | ív | 14 ft |
| 135g műnyomó papír B1 | ív | 25 ft |
| 135g volumenizált műnyomó papír A1 | ív | 15 ft |
| 135g volumenizált műnyomó papír B1 | ív | 15 ft |
| 135g volumenizált műnyomó papír C1 | ív | 15 ft |
| 150g műnyomó papír B1 | ív | 38 ft |
| 160g color copy SRA3 | ív | 13 ft |
| 175g fényes öntapadó papír B2 | ív | 52 ft |

| Megnevezés | Kiszerezés | Egységár | Darab |
|----------------------------------|------------|----------|-------|
| 115g volumenizált műnyomó p... | ív | 30 ft | 4 db |
| 1200g szűrkelemes karton B1 | ív | 150 ft | 5 db |
| 120g barna csikos kraft papír B1 | ív | 35 ft | 3 db |
| 120g color copy SRA3 | ív | 8 ft | 5 db |

32. ábra

Megrendeléshez eszközök összeállítása

A megrendelések teljesítéséhez szükséges eszközök kezelése az „Eszközök” menüpontban történik. „Beszállító” és „Admin felhasználó” jogosultsággal léphetünk be a menüpontba. A 33. ábrán látható táblázathoz hasonló kimenettel találkozunk itt, ahol fel vannak sorolva a rendelkezésre álló eszközök, a mennyiségi egységükkel és egységárjukkal. Az utolsó két oszlopban található ikonokkal tudjuk őket inaktívvá tenni vagy törölni. Az inaktív eszközöket nem lehet felhasználni megrendeléshez, ezeket a táblázat itt halványzürkével jeleníti meg.

Bejelentkezve: Superadmin Kilépés

Menü << **Eszközök**

Admin felhasználók Felhasználók Megrendelések **Eszközök**

Megnevezés Keresés Új eszköz hozzáadása

| ID | Megnevezés | Mennyiségi egység | Egységár | In/Aktiválás | Törés |
|----|------------------------------------|-------------------|----------|--------------|-------|
| 1 | teszt | ív | 25 Ft | - | x |
| 2 | 115g műnyomó papír B1 | ív | 20 Ft | - | x |
| 3 | 115g volumenizált műnyomó papír A1 | ív | 23 Ft | - | x |
| 4 | 115g volumenizált műnyomó papír B1 | ív | 30 Ft | - | x |
| 5 | 115g volumenizált műnyomó papír C1 | ív | 25 Ft | - | x |
| 6 | 1200g szürkelemez karton B1 | ív | 150 Ft | - | x |
| 7 | 120g barna csíkos kraft papír B1 | ív | 35 Ft | - | x |
| 8 | 120g color copy SRA3 | ív | 8 Ft | - | x |
| 9 | 120g fehér csíkos kraft papír B1 | ív | 35 Ft | - | x |
| 10 | 120g ofszet papír A1 | ív | 14 Ft | - | x |
| 11 | 135g műnyomó papír B1 | ív | 25 Ft | - | x |
| 12 | 135g volumenizált műnyomó papír A1 | ív | 15 Ft | - | x |
| 13 | 135g volumenizált műnyomó papír B1 | ív | 15 Ft | - | x |
| 14 | 135g volumenizált műnyomó papír C1 | ív | 15 Ft | - | x |
| 15 | 150g műnyomó papír B1 | ív | 38 Ft | - | x |
| 16 | 160g color copy SRA3 | ív | 13 Ft | - | x |
| 17 | 175g fényes öntapadó papír B2 | ív | 52 Ft | - | x |
| 18 | 175g matt öntapadó papír B2 | ív | 15 Ft | - | x |
| 19 | 180g ofszet papír B1 | ív | 52 Ft | - | x |
| 20 | 180g regiszter karton B1 | ív | 55 Ft | - | x |

Oldal 1 a 6-ból 20 Felhasználható eszközök: 1 - 20/119

33. ábra
Felhasználható eszközök listája

A 34. ábrán látható űrlap segítségével tudunk új eszközt rögzíteni vagy módosítani a már meglévők közül. Meg kell adnunk a nevét, a mennyiségi egységét illetve az árát.

115g volumenizált műnyomó papír B1

Megnevezés:

Mennyiségi egység:

Egységár: Ft

34. ábra
Eszköz űrlap

ÖSSZEGZÉS

Úgy gondolom, az elkészült programmal sikerült bemutatni, hogy hogy lehet keretrendszer segítségével webalkalmazást fejleszteni. A keretrendszer kiváló eszköz azoknak, akik ismerik a PHP nyelvet, és tisztában vannak az objektumorientáltsággal. A keretrendszer használata megszokást igényel, de ha egyszer a fejlesztő ráérez a használatára, akkor mindenképp hasznos eszköz a fejlesztéshez. Úgy gondolom, hogy minél nagyobb egy projekt, és minél többen vesznek részt benne, annál több időt és erőforrást tudnak spórolni azzal, ha keretrendszert használnak.

Láthatjuk, hogy a Yii Framework egy robosztus, a legtöbb területet – amire egy webes alkalmazás készítése során szükségünk lehet – érinti, sőt, kész mintákat kínál a használatukra. Ugyanakkor szabad kezet is ad a fejlesztőnek, saját magunk is implementálhatunk eszközöket és használhatjuk őket az előre adottak helyett.

Az elkészült alkalmazás jelenlegi formájában még nem alkalmas az éles használatra, de megfelelt annak a célnak, hogy bemutassam rajta keresztül a fejlesztés menetét. Számos bővítési lehetőséghez már elő van készítve, így akár néhány hetes fejlesztéssel alkalmassá lehet tenni arra, hogy kiváltsa az eredeti programot, amit a bevezetőben említettem. Magának az alkalmazásnak a működését másodlagosnak tekintettem, a hangsúlyt a bemutatandó technológiákra akartam helyezni.

IRODALOMJEGYZÉK

1. WIKIPEDIA QBASIC. [Online] 2012. [Hivatkozva: 2012. 04 25.]
<http://en.wikipedia.org/wiki/QBasic>
2. WIKIPEDIA PASCAL (*programming language*). [Online] [Hivatkozva: 2012. 04 25.]
[http://en.wikipedia.org/wiki/Pascal_\(programming_language\)](http://en.wikipedia.org/wiki/Pascal_(programming_language)).
3. INTERNET WORLD STATS [Online] [Hivatkozva: 2012. 04 25.]
<http://www.internetworldstats.com/eu/hu.htm>.
4. WIKIPEDIA COMPARSION OF WEB APPLICATION FREMAWORKS [Online] [Hivatkozva: 2012. 04 25.]
http://en.wikipedia.org/wiki/comparsion_of_web_application_frameworks.
5. DAVID CONELLY: THE BEST PHP FRAMEWORK OF 2011 [Online] [Hivatkozva: 2012. 04 25.] <http://davidjconelly.wordpress.com/2011/07/03/the-best-php-framework-of-2011>.
6. PHP FRAMEWORKS. [Online] [Hivatkozva: 2012. 04 25.]
<http://www.phpframeworks.com/>.
7. YII FRAMEWORK EXTENSIONS [Online] [Hivatkozva: 2012. 04 25.]
<http://www.yiiframework.com/extensions>.
8. YII FRAMEWORK TUTORIALS [Online] [Hivatkozva: 2012. 04 25.]
<http://www.yiiframework.com/tutorials/>.
9. GITHUB YII MASTER [Online] <https://github.com/yiisoft/yii/commits/master>.
10. WIKIPEDIA MODEL-VIEW-CONTROLLER [Online] [Hivatkozva: 2012. 04 25.]
<http://en.wikipedia.org/wiki/model-view-controller>.
11. YII FRAMEWORK ABOUT YII [Online] <http://www.yiiframework.com/about>.
12. YII FRAMEWORK YII 1.0.0 IS RELEASED [Online] [Hivatkozva: 2012. 04 25.]
<http://www.yiiframework.com/news/3/yii-1-0-0-is-released>.
13. OPEN SOURCE INITIATIVE: BSD LICENSE [Online] [Hivatkozva: 2012. 04 25.]
<http://www.opensource.org/licenses/bsd-license.php>.

14. WIKIPEDIA BSD LICENC [Online] [Hivatkozva: 2012. 04 25.]
<http://hu.wikipedia.org/wiki/BSD-licenc>.
15. YII FRAMEWORK PERFORMANCE OF YII [Online] [Hivatkozva: 2012. 04 25.]
<http://www.yiiframework.com/performance>.
16. WIKIPEDIA MODELL-NÉZET-VEZÉRLŐ [Online] <http://hu.wikipedia.org/wiki/Modell-nézet-vezérlő>.
17. APPLICATIONS PROGRAMMING IN SMALLTALK-80: HOW TO USE MODEL-VIEW-CONTROLLER [Online] <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>.
18. WIKIPEDIA CREATE, READ, UPDATE, DELETE [Online]
http://en.wikipedia.org/wiki/Create,_read,_update_and_delete.
19. YII FRAMEWORK MODEL-VIEW-CONTROLLER [Online] [Hivatkozva: 2012. 04 27.]
<http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc>.
20. XUE, QIANG: BUILDING A BLOG SYSTEM USING YII 2008.
21. ZHUO, QIANG XUE AND XIANG WEI: THE DEFINITIVE GUIDE TO II 1.1. 2008.
22. ZEND FRAMEWORK PROGRAMMER'S REFERENCE GUIDE [Online]
<http://framework.zend.com/manual/en/coding-standard.html>.
23. SEBESTA, ROBERT W. A WORLD WIDE WEB PROGRAMOZÁSA. Budapest : Panem Kiadó, 2005.
24. MOULDING, PETER: PHP HALADÓKNAK FEKETE KÖNYV. Budapest: Perfect-Pro Kft., 2002, old.: 128-129.
25. NETBEANS [Online] <http://netbeans.org>.
26. PHPMYADMIN [Online] <http://www.phpmyadmin.net>.
27. WEBYOG [Online] <http://www.webyog.com>.
28. FIREFOX ADDONS: FIREBUG [Online]
<https://addons.mozilla.org/hu/firefox/addon/firebug/statistics/?last=365>.
29. LOGOUT.HU: VERZIÓKÖVETÉS [Online]
http://logout.hu/cikk/verziokovetes/a_verziokovetes_lenyege.html.
30. GOOGLE CODE [Online] <http://code.google.com/intl/hu-HU/>.

31. GITHUB [Online] <https://github.com/>.
32. ATlassian BITBUCKET [Online] <https://bitbucket.org/>.
33. WIKIPEDIA MD5 [Online] <http://hu.wikipedia.org/wiki/MD5>.
34. SENCHA DOCS: EXTJS 4.0 [Online] http://docs.sencha.com/ext-js/4-0/#!/guide/application_architecture.
35. SENCHA: COMPONENTS [Online] <http://www.sencha.com/learn/components/>.
36. JQUERY IMPROMPTU [Online] <http://trentrichardson.com/Impromptu/>.
37. JQUERY VALIDATION [Online] <http://bassistance.de/jquery-plugins/jquery-plugin-validation/>

MELLÉKLETEK

TELEPÍTÉSI ÚTMUTATÓ

Követelmények

- Apache HTTP szerver
- PHP 5.2.1 vagy újabb
- MySQL szerver 5.1.33 vagy újabb
- Test Mail Server Tool (dvd-n mellékelve)
- Yii Framework (dvd-n mellékelve)
- Böngésző alkalmazás

A dvd-n mellékeltem a Wamp alkalmazást, ami biztosítja az Apache szervert, a PHP-t és a MySQL szervert is.

Szerver beállítások:

- A webszerveren engedélyezni kell a `httpd-vhosts-t` (a `httpd-conf-ban` lehet)
- A `httpd-vhosts-ba` tegyük be a `szakdolgozat/_doc/szakdolgozat.conf` tartalmát (a saját mappabeállításainkkal)
- A `hosts` fájlba tegyük be a következőket:
 - `127.0.0.1 szakdolgozat.local`
 - `127.0.0.1 admin.szakdolgozat.local`
- A PHP-ben engedélyezni kell a `short_open_tag-et`
- Hozzuk létre az adatbázist a `szakdolgozat/_doc/adatbazis.sql`-ből

A telepítés lépései:

- Másoljuk be a webszerver mappánkba a DVD-n található `szakdolgozat` és `yii_1_8` mappákat
- Böngészőből hívjuk meg a `<webszerver>/yii_1_8/requirements` címet a Yii ellenőrzéséhez
- Állítsuk be a `protected/config/custom.php-t` a `szakdolgozat/_doc/custom.php.example` alapján, a következő paramétereket állítsuk be:
 - `mainPath`

- sitePath
 - adminPath
 - siteUrl
 - adminUrl
 - adminEmail
- Adatbázis kapcsolathoz:
 - connectionString
 - username
 - password
- Adjunk írási jogot a `<webszerver>/szakdolgozat/protected/runtime` és a `<webszerver>/szakdolgozat/site/assets` mappáknak

Ha mindent jól beállítottunk, akkor a `szakdolgozat.local` címen fog elindulni az oldal, az admin modul pedig az `admin.szakdolgozat.local` címen.

Adminisztrátor modul használatához felhasználónév és jelszó: *superadmin*.

Megrendelő oldalhoz tesz felhasználó és jelszó: *teszt@teszt.hu, 12345*

Demo alkalmazás

A demo alkalmazás a dolgozat nyomtatásakor még nem volt elérhető, így ennek elérhetőségéről kérem, érdeklődjön a *gobl.adi@gmail.com* e-mail címen.