Tartalom

[Összefoglalás 1](#_Toc386447957)

[Bevezetés 2](#_Toc386447958)

[Témaválasztás 2](#_Toc386447959)

[Eredeti specifikáció 3](#_Toc386447960)

[Tervezés 4](#_Toc386447961)

[Weboldalak elkészítésének lehetőségei 5](#_Toc386447962)

[Fejlesztői környezet 6](#_Toc386447963)

[Nyelvek kiválasztása 6](#_Toc386447964)

[PHP 6](#_Toc386447965)

[MySQL 8](#_Toc386447966)

[Apache 9](#_Toc386447967)

[Framework-ök 10](#_Toc386447968)

[Fat-free 10](#_Toc386447969)

[Twitter bootstrap 16](#_Toc386447970)

[Az elkészült program 17](#_Toc386447971)

[Áttekintés 17](#_Toc386447972)

[Az interface 17](#_Toc386447973)

[Megvalósult dolgok, kihagyott funkciók 17](#_Toc386447974)

[Adatbázis 18](#_Toc386447975)

[Adatbázis elérése PHP-ból 21](#_Toc386447976)

[Biztonság 23](#_Toc386447977)

[Felhasználók azonosítása és kezelése 23](#_Toc386447978)

[A jelszavak titkosítása 23](#_Toc386447979)

[A felhasználók beléptetése 24](#_Toc386447980)

[Tesztelés 24](#_Toc386447981)

[Osztályok, mappák tartalmának részletesebb elemzése 25](#_Toc386447982)

[Felhasználói dokumentáció 28](#_Toc386447983)

[Oldal felépítése 28](#_Toc386447984)

[Főoldal 28](#_Toc386447985)

[Regisztrációs oldal 28](#_Toc386447986)

[Belépés 29](#_Toc386447987)

[Üzenetek 29](#_Toc386447988)

[Keresés 29](#_Toc386447989)

[Profil 29](#_Toc386447990)

[Privát Üzenet 30](#_Toc386447991)

[Kommentelés 30](#_Toc386447992)

[Hirdetés feladása 30](#_Toc386447993)

[Hirdetéseim 31](#_Toc386447994)

[Hirdetés-oldal 31](#_Toc386447995)

[Github 32](#_Toc386447996)

[Összefoglalás 34](#_Toc386447997)

[Telepítés és futtatás 35](#_Toc386447998)

[Irodalomjegyzék 36](#_Toc386447999)

# Összefoglalás

Szakdolgozat egy Apród/Hardverapróhoz hasonló weboldalról, PHP, HTML, CSS alapokon. Ahol a felhasználók könnyen és gyorsan értékesíthetik használt/megunt tárgyai

# Bevezetés

## Témaválasztás

A mai világban az internet széles körben elterjedt, az emberek életének a szerves részévé vált. A legtöbb felhasználó böngészőn keresztül éri el az internetet, amely ma már minden számítógépen megtalálható és a mobiltelefonok nagy részén is. Ez egy olyan multifunkcionális eszközzé fejlődik, amelyen keresztül a felhasználók információt gyűjthetnek, kommunikálhatnak, filmeket nézhetnek. A kis és nagy vállalatok számára a marketing mind az értékesítés területén elengedhetetlen eszköz. Szóval a jövőnek mindenképpen szerves része és egy olyan eszköz amire a fejlesztést mindenképpen érdemes megtanulni és hosszútávon egy jó befektetés.

Mivel teljesen lenyűgözött ez az eszköz, úgy döntöttem a nyáron hogy szeretnék én is foglalkozni vele, később beiratkoztam a Jedlikbe, és amikor a szakdolgozat témaválasztására került a sor a választás nem volt túl nehéz, valami olyat szerettem volna fejleszteni aminek a témája komolyabb mint egy játék. Később munkakeresésnél referenciának vagy egyetemen valamilyen formában újrahasznosítható. Nem utolsó sorban szakmailag kihívás és lehetőséged ad ezen a területen való fejlődésre.

Úgy gondoltam, hogy a program logikai rész megírására a PHP-t használom mivel elég gyorsan lehet vele fejleszteni (dinamikus változók, megengedő syntax). Az oldal megformázásához HTML, CSS-re esett a választás, de erről majd később.

## Eredeti specifikáció

* Szakdolgozatnak egy piacteret fogok készíteni (Vatera, eBay).
* Megjelenítés: html, css, javascript. Szerveroldalon: PHP, MySQL adatbázissal.
* A felhasználóknak regisztrálniuk kell (email-es megerősítés).
* Főoldalon véletlenül kilistázott termékek, kategóriák.
* A felhasználók termékeket (hirdetéseket) tudnak feltölteni címszavakkal, képpel, kategória, leírással.
* Keresés: címszavakra rálehet keresni, kilistázza a találatokat.
* Szűrés bizonyos opciókra (szállítás, ár…).
* A termékeknek külön kigenerált oldala lesz:
  + Termékleírás
  + Kép
  + Termék helye
  + Hirdetés érvényessége
  + Kommentek
  + Hirdető profilja
* Az üzlet lebonyolítása után lehet értékeli egymásnak a feleket.

# Tervezés

Az oldal tervezésénél az volt az elsődleges szempont hogy egyszerű legyen és könnyen használható az egyszerű felhasználók számára is. Ez azért fontos, mert egy jól kialakított aukciós oldal szélesebb körben válik ismertté, népszerűvé, remélhetőleg ennek függvényében az oldal használóinak száma is nőni fog. Ellenkező esetben, ha egy oldal lassú vagy nem lehet tájékozódni rajta, a felhasználók inkább továbbállnak és keresnek egy másik oldalt a témában.

Ehhez a következő tényezőket kell figyelembe venni:

* Fontos a gyorsaság. Minél hamarabb tölti le az oldalt a felhasználó, annál élvezetesebb lesz számára a böngészés. A sebességet sok tényező befolyásolja, a kiszolgáló szerver távolsága, teherbírása, sebessége, a felhasználó sávszélessége.
* Az oldal kinézete ne menjen a gyorsaság rovására. Legfontosabb az információ szerzése, ehhez az információhoz kell minél hamarabb hozzájuttatni a felhasználót. Ennél az oknál fogva nem használok flash animációkat, nagyméretű képeket, különféle grafikus elemeket. A jól kiépített, könnyen használható felületnek nagyobb hasznát veszik a felhasználók, mint egy túl díszített oldalnak.
* Css használatával egyszerűbb egységes oldalakat készíteni, és változtatásokat is könnyebben lehet eszközölni.
* Modern megjelenés.
* Lehetőleg minden böngésző program azonos formában jelenítse meg az oldalt. Kisebb eltérések persze adódnak, de ez az oldal használhatóságát nem csorbíthatja.
* Sütik (cookie) engedélyezése nélkül is teljes funkcionalitással működjön az oldal.
* Navigáció: Egy felhasználó, ha az oldalhoz ér, könnyen fel tudja térképezni, mi hol van, mire kell kattintani. A legfőbb cél, hogy a felhasználó az adott információt a lehető legrövidebb időn belül megtalálja.
* Javasolt a legtöbb gépen lévő alapértelmezett fontok használata, mivel ezek általában könnyen olvasható betűk.
* Kerülendő a rikító, szemet fárasztó színösszeállítást.

## Weboldalak elkészítésének lehetőségei

Alapvetően két lehetőségünk van honlapunk elkészítésére. Az első, hogy forráskódot szerkesztünk. Ennek előnye, hogy megfelelő programozói felkészültséggel bármilyen igényeket kielégítő oldalakat készíthetünk. Ehhez azonban rengeteg időre van szükségünk. Már az is sok időt vesz igénybe, hogy megtanuljuk a szükséges programnyelvek szintaktikáját, nem is beszélve a konkrét feladat elkészítéséről. Az elkészült weboldal tesztelése során felmerülő hibák javítása sem egyszerű feladat. Második lehetőségünk, hogy kész programokat használunk honlapunk elkészítéséhez. Ezek az alkalmazások nem igényelnek programozói szaktudást, szinte bárki dolgozhat velük, ha rendelkezik általános számítástechnikai ismeretekkel. A magyar nyelvben ezeket a rendszereket tartalomkezelő rendszereknek nevezzük, általánosságban pedig CMS rendszerekként szokták emlegetni, ami az angol Content Managament System szavak kezdőbetűiből származik. Mivel az én szakdolgozatom egy programozói szakdolgozat ezért a CMS rendszer szóba sem jöhetett ezért a dolgok nagy részét kézzel írtam.

# Fejlesztői környezet

A programkódok nagy részét a Netbeans IDE-ben írtam meg. (PHP, CSS, HTML)

Az adatbázis kialakításához és menedzseléséhez a Navicat-ot használtam.

Helyi hálózaton való teszteléshez az Easy PHP csomagját használtam:

* Apache
* MySQL

Teszteléshez a google chrome-t használtam a megjelenés erre van letesztelve, optimalizálva.

## Nyelvek kiválasztása

Egy piactér szerű dinamikus oldal fejlesztéséhez elengedhetetlen egy jó programnyelv választása nálam a PHP-ra esett a választás. A böngészőben való megjelenítéshez a leíró nyelvek közül nem ekkora a választék, ezért a két legelterjedtebbre esett a választás a HTML5 és a CSS3-ra.

## PHP

A programlogika (buisness logic) megírása ezen a nyelven folyik. A legújabb változatot használom mivel ez már támogatja az objektum orientált programozást és az új függvények is hasznosak. A szintaxisa megengedő nem kis-nagy betű érzékeny és néhány parancs többször is szerepel különböző néven, ezenkívül az összes változó dinamikus. Ezek tulajdonságok lehetnek ugyanúgy előnyök, mint hátrányok mivel nagy szabadságot ad a programozóknak könnyedén lehet csúnya és átláthatatlan kódot írni. Ezeket figyelembe véve próbáltam valami egyszerű konvenciót kialakítani hogy az eltérő időben megírt kódok egyformán nézzenek ki. A programot szinte teljes egészében objektumorientáltan írtam meg mivel szerintem ez átláthatóvá teszi a kódot és többszemélyes fejlesztés, nagyobb projektek esetén az procedurális programkód szóba sem jöhet.

A PHP (Hypertext Preprocessor) 1995-ben jelent meg. 1994-ben Rasmus Lerdorf tervezte, de a további fejlesztést már a PHP Group végzi. De Facto szabvány, nincs hivatalos specifikációja. Dinamikus weboldalak készítéséhez használatos, nyílt forráskódú, szabad szoftver. Imperatív, objektumorientált szkriptnyelv. Széles körben alkalmazható, mivel platform független. Mivel Lerdorfnak eredetileg néhány Perl szkript lecserélése volt a célja, ezért a PHP-ben megtalálhatók belőle bizonyos vonások, mint például a változók. A PHP programokat általában HTML oldalba építve futtatják, ritkán HTML nélkül, parancssoros programként is alkalmazzák. A következő három területen alkalmazzák legfőbbképpen a PHP-t:

- Szerver oldali programozás: ez a leghagyományosabb és legfőbb területe a PHP alkalmazásának. Itt három fő dologra lesz szükségünk: egy PHP elemzőre / értelmezőre, egy web szerverre, és egy böngészőre. A web szervert egy megfelelően beállított PHP-vel kell futtatnunk. A szerver által és a böngésző segítségével tekinthetjük meg a PHP program eredményét, kimenetét. Mindezt elvégezhetjük az otthoni gépünkön is.

- Parancssoros programozás: szerver és böngésző nélkül is készíthetünk így PHP alkalmazásokat, mindössze a PHP értelmezőre lesz szükség. Ez hasznos lehet például feladatok ütemezésére, vagy akár egyszerű szövegfeldolgozásra.

- „Grafikus” alkalmazások: itt a grafikus szó azt jelentené, hogy a program rendelkezik egy grafikus felhasználói felülettel. Habár a PHP nem a legmegfelelőbb nyelv ilyen alkalmazások készítésére, de ha már elég jól ismerjük a nyelvet, és használni szeretnénk a PHP fejlettebb képességeit a kliens-oldali alkalmazásainkban, akkor a PHP-GTK lesz segítségünkre.

A PHP további területeken is széleskörű választási lehetőségeket biztosít: - Operációs rendszerek: a PHP a legtöbb, illetve a legelterjedtebb operációs rendszereken működtethető, mint például: Linux, sok Unix változaton (HP-UX, Solaris, OpenBSD), Microsoft Windows, Mac OS X, RISC OS, és még valószínűleg sok máson is.

- Web szerverek: ma a PHP már a legtöbb web szervert támogatja, mint például: Apache, Microsoft Internet Information Server, Personal Web Server, Netscape és iPlanet szerverek, Oreilly Website Pro, Caudium, Xitami, OmniHTTPd, és még sok mást is. A PHP CGI feldolgozóként is képes működni. PHP programozáshoz készítettek különböző fejlesztői eszközöket, melyek megkönnyítik a programozó munkáját, de PHP program megírható a legegyszerűbb szövegszerkesztővel is.

Egy egyszerű példa a HTML-be ágyazott PHP-ról:

**<html>**

**<head></head>**

**<body>**

**<?php**

**echo ”Helló Web!”;**

**?>**

**</body>**

**</html>**

Tehát PHP kódot HTML kódban bárhol elhelyezhetünk. PHP kód kezdetét a következőképpen jelezhetjük: <?php illetve <? , PHP kód végét pedig ?> -el.

A PHP lehetőséget biztosít arra, hogy adatbázissal kapcsolatot hozzunk létre. PHP programok esetében elmondható, hogy leginkább a MySQL adatbázis-kezelőt szokták használni.

## MySQL

A MySQL egy több felhasználós, többszálú, SQL-alapú relációs adatbázis-kezelő szerver. Az MySQL az egyik legelterjedtebb adatbázis-kezelő, aminek egyik oka lehet, hogy a teljesen nyílt forráskódú LAMP (Linux–Apache–MySQL–PHP) összeállítás részeként költséghatékony és egyszerűen beállítható megoldást ad dinamikus webhelyek szolgáltatására. A MySQL honlapjáról grafikus felületű adminisztráló eszközök is letölthetők: MySQL Administrator és MySQL Query Browser. Széles körben elterjedt és népszerő alternatíva a PHP nyelven írt, nyitott forráskódú phpMyAdmin. A phpMyBackupPro (amelyet szintén PHP-ban írtak) adatbázisok (akár időzített, ismétlődő) mentésére szolgál eszközül. A phpMyAdmin a MySQL adatbázisok WEB alapú, teljes körű adminisztrációját teszi lehetővé. Én a Navicat-et használtam az adatbázis kezelésére, mivel szerintem ez sokkal jobb mint a többi ingyenes program.

## Apache

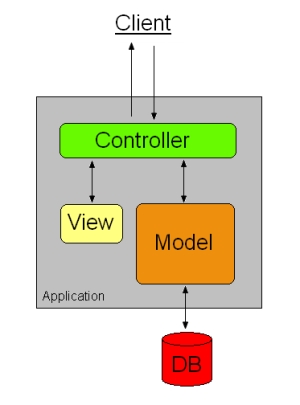
Az egyik legelterjedtebb nyílt forrású, szabadon terjeszthető web szerver, amelynek különböző változatai többek között Windows, Linux operációs rendszerek alatt is futtathatóak.

# Framework-ök

## Fat-free

A projekt programozás részének október körül álltam neki de rövidesen rájöttem hogy a programkód és a megjelenítés összevegyítése nem a legcélravezetőbb dolog hosszútávon. Ekkor a programozást egy kis időre abbahagytam és jött a kutatás rész, ekkoriban botlottam az „MVC” elnevezésbe ami a Modell-View-Controller rövidítése, ennek az a lényege hogy az oldalt 3 részre bontjuk:

* **Modell**: Itt történnek a számítások, a vezérlőtől megkapott adatok feldolgozása az adatbázissal való kommunikáció, sajnos nálam ez néhol egybeolvadt a vezérlővel a gyorsabb fejlesztés érdekében.
* **Nézet**: ezen a felületen történik a felhasználói interakció, jellemzően egy felhasználói felület. Különböző célokra különböző nézetek létezhetnek ugyanahhoz a modellhez.
* **Vezérlő**: Ez a réteg köti össze a modell-t a nézettel. A bemeneti adatok ezen keresztül jutnak el a modellhez amely azt feldolgozza. A feldolgozott adatot ez adja át a nézetnek.



1. ábra: Az MVC minta egy általános ábrázolása

Ez után tovább keresgéltem a megoldás után mivel ezt nem igazán tudtam elképzelni hogy hogyan programozzam le, ekkor az egyik ismerősöm ajánlotta hogy használjak egy PHP frameworkot, így esett a választás a Fat-Free frameworkre (késöbbiekben csak F3). Ez egy kisebb framework ami támogatja az MVC fejlesztési mintát. Legfontosabb funkciói:

**URL routing**: Ez az összes url-t az index.php ra irányítja egy HTACCESS file segítségével ahol aztán defeniálva van minden url-nek egy funkció ami annak a kérésnek megjelenít egy oldalat vagy elvégez egy feladatot. az ilyen útvonalak lehetnek: staikusak és dinamikusak.

1-1 példa mindkettőre:

* **Statikus**:

$f3->route('GET /Register', 'RegisterController->display');

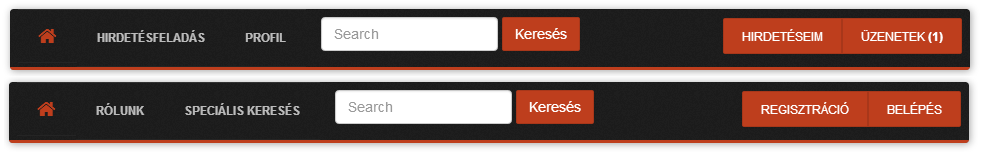
Amennyiben a felhasználó beírja az oldal nevét/register (ez alap esetben GET (request method)) meghívja a RegisterController osztály display funkcióját amely megjeleníti a regisztrációs oldalt.

* **Dinamikus**:

$f3->route('GET /Ad/@adid','AdController->specificad');

Amikor a felhasználó oldalneve/Ad/tetszőleges számot ad meg, meghívódik az AdController osztály specificad funkciója amely eldönti hogy létezik e ilyen ’id’-vel rendelkező hirdetés és ennek függvényében megjeleníti a hirdetést vagy a hibaüzenetet.

Amennyiben a routing listában nem definiált oldalt hívunk egy 404 oldalra dob.



2. ábra: A navbar belépett & nem belépett felhasználók számára

**Template kezelő rendszer:** Számomra ez a funkció is nagyon hasznosnak bizonyult ez különíti el a nézetet a logikai résztől. Az előző példák közül az első:

function display($f3) {

NAVBARController::buttons($f3);

echo Template::instance()->render('main.tpl');

echo Template::instance()->render('register.tpl');

echo Template::instance()->render('endofmain.tpl');

}

Ez egy elég egyszerű funkció, az első sora (NAVBARController::buttons($f3);) meghív egy button nevű funkciót ami beállítja az oldal tetején lévő gombok értékeit a attól függően hogy a felhasználó bevan-e jelentkezve. Ebből egy részlet:

if (isset($\_SESSION['username'])) {

$f3->set('signinstatus', 'Hirdetésfeladás');

$f3->set('signinstatuslink', '/adupload');

} else {

$f3->set('signinstatus', 'Rólunk');

$f3->set('signinstatuslink', '/aboutus');

}

A set nevű funkció beállít framework változókat bizonyos értékekre.

A display funkció következő sora:

echo Template::instance()->render('main.tpl');

kirendereli a main nevezetű templatet:

<!DOCTYPE html>

<html>

<head>

<title>Piactér</title>

<include href="style.tpl" />

</head>

<body>

<include href="navbar.tpl" />

<div class="wrapperSJ" id="content">

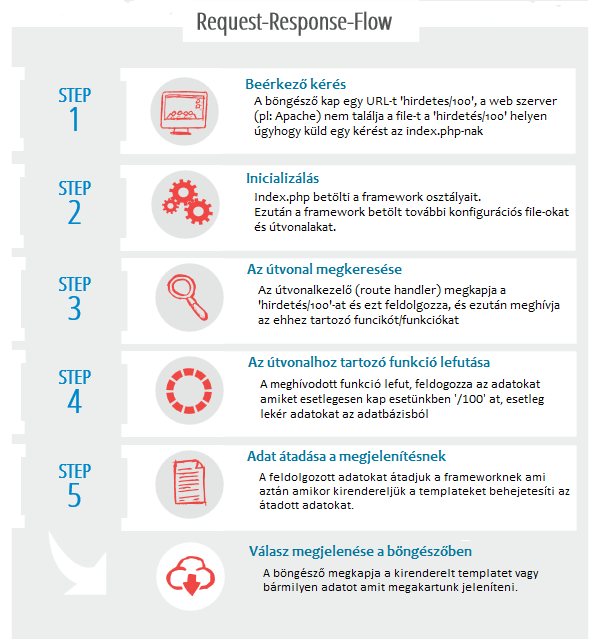
<div class="row">

<!-- END OF MAIN TPL -->

Ez a rövid részlet lesz az oldal forrásának az eleje, egy hozzáértő ember rögtön észreveszi, hogy ebben van kettő darab nem ide illő tag: <include href="style.tpl" /> és <include href="navbar.tpl" />. Az include tageket a template kezelő rendszer értelmezi és a megadott templateket beilleszti a helyére. Az első include a css-eket és javascripteket illeszti be. A második pedig a navbart, ebben találhatóak framework változók, pl:

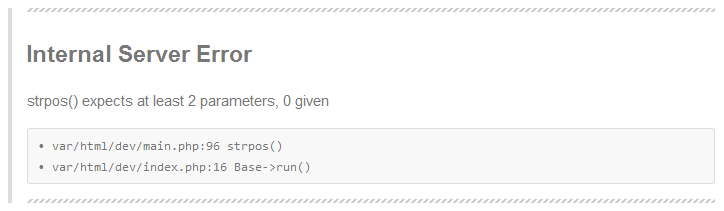
* {{ @signinstatus }}
* {{ @signinstatuslink }}

Ezeknek az értékét állította be a NAVBARController::buttons($f3) úgyhogy itt a bejelentkezett felhasználók, a hirdetésfeladás gombot kapják „/adupload” linkkel és nem bejelenkezettek pedig egy Rólunk feliratú gombot kapnak „/about us” linkkel.

**

3. ábra: A fatfree kérés-válasz folyamata.

**Autoloader** egy framework által kezelt tömb amiben mappanevek vannak eltárolva ennek köszönhetően a funkciókban nem kell require/incude-olni az osztályokat amelyeket használunk. Amikor osztályt példányosítunk vagy statikusan hívunk függvényt automatikusan megnézi a megadott mappákat és amennyiben ugyanolyan néven vannak elmentve a file-ok mint a bennük tárolt osztály (csak egy osztály lehet egy fileban) az autoloader megtalálja őket és meghívja őket helyettünk.

**Hibakezelő rendszer** az F3 hibakezelő rendszere a hibákat egy saját maga által kigenerált HTML oldalon jeleníti meg ezáltal is gyorsabbá téve a fejlesztést:

4. ábra: Hibaüzenet

A Framework rendelkezik egy debug nevű változóval ezzel lehet állítani a hibákat kiváltó források elemzésének a részletességét. 0-3 ig lehet állítani a nagyobb a szám a részletesebb leírás ez a változó nagyon hasznos mivel fejlesztésnél szükséges az összes hiba kiírása a hatékony debuggoláshoz míg amikor a program/oldal az interneten „élesben” fut ugyanez komoly biztonsági réseket fedhet fel a hozzáértők előtt. Ezért amikor a fejlesztő feltölti a termékét érdemes ezt a változót 0-ra állítani így az esetlegesen megjelenő hibákat kiváltó okok rejtve maradnak a felhasználók előtt.

## Twitter bootstrap

Mivel az oldal vizuális megtervezésre, megvalósítására nem akartam túl sok időt, eltölteni ezért úgy döntöttem, hogy kipróbálom a Twitter-nek az „új” frontendes keretrendszerét. Ez tulajdonképpen kettő css file, az egyik az oldal elrendezését „layout” biztosítja hogy mindig egyforma maradjon, a másik pedig egy „skin” ami az előző rétegnek színt, árnyékokat, egy modern megjelenést ad. Nagyon fontos hogy minden második rétegben szerepelnek az első réteg elemei ebben a listában egy átlagos weboldal felépítéséhez minden szükséges elem megtalálható, input mezők, gombok, stb… Ezenfelül pedig általában minden fejlesztő ad hozzá néhány sajátot. Az egységesség azért fontos hogy az oldal megjelenését könnyen és gyorsan meglehessen változatni.

# Az elkészült program

## Áttekintés

Az oldal az előzőekben leírt keretrendszerek és program/leírónyelvek használatával készült.

## Az interface

A megjelenítés mindig 3 részből tevődik össze:

* **Navbar**, az oldal tetején helyezkedik el,ez tartalmazza a navigációhoz szükséges gombokat, logót, közösségi oldalra linkeket.
* **Tartalom**, ez tartalmazza mindig az éppen aktuális tartalmakat amire a felhasználó kíváncsi, itt történik az interakció, adatcsere nagy része.
* **Footer**, a lap alján található, az oldalról alapvető információk…

## Megvalósult dolgok, kihagyott funkciók

A specifikációban leírt dolgoknak a nagy része megvalósult, azokkal a nyelvekkel sikerült elkészítenem amelyeket az elején elgondoltam.

Amik nem valósultak meg:

* Email-es megerősítés regisztrációnál
* A hirdetéseknek nincsen kategóriájuk, nem igazán van kialakított hely címszavaknak
* Címszavakra rálehet keresni, kilistázza a találatokat.
* Szűrés bizonyos opciókra (szállítás, ár…)

**Email-es megerősítés regisztrációnál:** Ennek a funkciónak kihagyásának a legnagyobb oka volt hogy úgy éreztem ez nem egy olyan probléma ami programozás terén nagyobb kihívás. Itt az email szerver beállítása jelentette volna a legnagyobb akadályt.

**A hirdetéseknek nincsen kategóriájuk, nem igazán van kialakított hely címszavaknak:** Ez nem egészen igaz mivel a hirdetéseknek belehet állítani állapotot (Új/Használt), helységet hogy hol található meg, de a hagyományos értelemben vett kategóriák az oldalon nem léteznek, ez főleg azért maradt ki mivel kicsit másabb irányba ment el a fejlesztés és a végén már nem maradt rá idő.

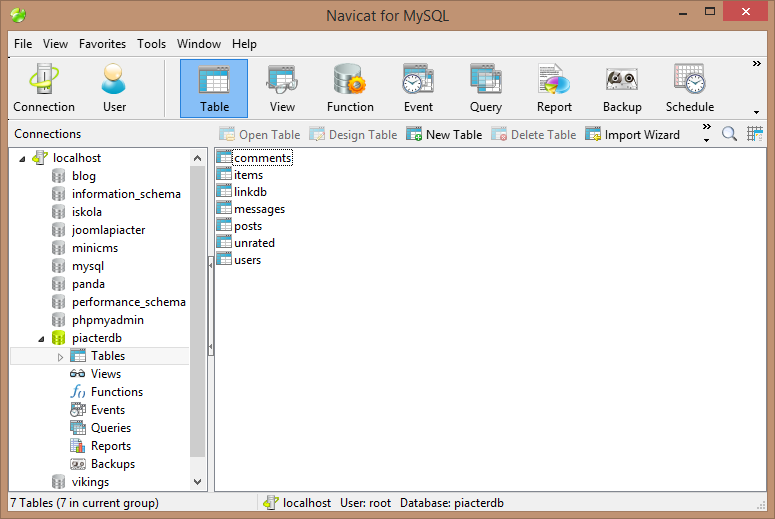
**Címszavakra rálehet keresni, kilistázza a találatokat. Szűrés bizonyos opciókra (szállítás, ár…):** Ezek a funkciók azért maradtak ki az oldalból mivel nem volt időm lefejleszteni őket.

## Adatbázis

„A MySQL az egyik legelterjedtebb adatbázis-kezelő, aminek egyik oka lehet, hogy a teljesen nyílt forráskódú LAMP (Linux–Apache–MySQL–PHP) összeállítás részeként költséghatékony és egyszerűen beállítható megoldást ad dinamikus webhelyek szolgáltatására.”

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

Azért esett a választásom a MySQL-re mivel ez az egyik legelterjedtebb adatbázis-kezelő, ami azt jelenti hogy ehhez van a legtöbb leírás, dokumentáció, valamint ha hiba merül fel több ember segítségére számíthatok.



5. ábra: Adatbázis

Táblák részletezése:

* **Comments:** Ebben a táblában tárolom a Hirdesekhez való kommenteket
* **Items:** Itt vannak a felhasználók által feltöltött hirdetések adatai, lejárat, cím, kép neve.
* **Linkdb:** Ez egy teszt tábla a keresés kipróbálására mivel rengeteg idő lenne annyi adatot felvinni hogy rendesen kipróbálható lehessen az items táblából. Ennek a táblának a tartalma a Vateráról lett letöltve egy kis script segítségével.
* **Messages:** Itt vannak az üzenetek amiket a felhasználók egymásnak küldenek valamint azok az üzenetek is amiket az oldal küld a hirdetéssel kapcsolatban (sikeres licit stb…)
* **Posts:** Ez is egy teszttábla, jelenleg nem használom semmire.
* **Unrated:** Ebben a táblában vannak eltárolva a felhasználók értékelései egymás felé.
* **Users:** Itt van az összes felhasználó személyes adata, email, jelszó(titkosítva), legutolsó belépés...

Nagy általánosságban igyekeztem optimalizálni az adatbázist, hogy semmilyen adat ne legyen többször eltárolva és mindent egyedi azonítóval érjek el ha egy másik táblában szükség van rá.

**Az eventek** a mysql 5.1.6-os verziójában jelentek meg. A cron jobok és az időzített feladatok mellé egy alternatíva. A lényege hogy egy bizonyos időben valamilyen feladatot eltudnak végezni (UPDATE, DELETE, INSERT…). A projektemben az aukciót lejárását oldottam meg velük, amennyiben egy felhasználó feltölt egy olyan hirdetést, ami egy bizonyos idő után lejár akkor, a szerver egy eventet generál ami pontosan akkor fut le amikor a hirdetés lejár. Ezzel rengeteg erőforrást spórolunk meg ahhoz képest mintha ezt pl cron jobokkal végeznénk el és sokkal pontosabb is. Persze teljes egészében nem tudja kiváltani ezeket az időzített feladatokat.

Ez a funkció akkor hívódik meg amikor a felhasználó egy olyan hirdetést ad fel amihez van aukció is. Létrehoz egy eventet ami akkor lefut amikor lejár az aukció. Később egy kis script amit bizonyos időközönként lefuttat a szerver, leellenőrzi hogy melyik hirdetés lett így lezárva kiküldi a leveleket a felhasználóknak (amennyiben volt nyertes) és hozzáad egy üres értékelést a két felhasználó profiljához.

private function auctionevent($f3) {

$sessid = $\_SESSION['id'];

$connection = new PDOConnection;

$result = $connection->query("SELECT availability, id FROM items WHERE owner='$sessid' ORDER BY date DESC ")

->fetchAll(PDO::FETCH\_ASSOC);

$expire = $result[0]['availability'];

$adid = 'AD' . $result[0]['id'];

$connection->query("CREATE EVENT IF NOT EXISTS $adid

ON SCHEDULE AT '$expire'

DO

UPDATE `items` SET isopen=0 WHERE owner='$sessid' AND availability='$expire';");

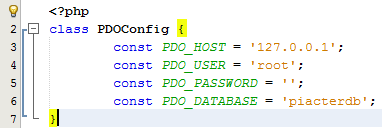
}

## Adatbázis elérése PHP-ból

Jelenleg a PHP-nak 3 API-ja van MySQL adatbázisok elérésére:

* **Mysql:** Ez a legrégebbi, még mindig sok helyen használják a 3 legnagyobb problémája:
  + **Elavult (Deprecated):** A PHP már hivatalosan érvénytelenítette így várható hogy valamelyik következő verzióban kiveszik a támogatást, annak ellenére hogy nagyon széles körben elterjedt mind a mai napig.
  + **Nem biztonságos:** Minden adatot átad egyenesen a fejlesztőnek így olyan karakterek, szövegrészletek is bennmaradhatnak amelyek biztonsági problémákat vethet fel a nem elég tapasztalt fejlesztők számára.
  + **Flexibilitás:** A megírt kódok csak MySQL adatbázissal működnek, jelentős hiányosságai vannak támogatás terén.
* **Mysqli:** Ez a mysql API-nak egy fejlesztett verziója de ebben is jelentős hiányosságok lelhetőek fel, főleg mert ez sem túl flexibilis. Sebesség szempontjából viszont ez a leggyorsabb.
* **Pdo:** A PDO nem más mint egy objektum az adatbázis kapcsolatok, lekérdezések, stb. kényelmes, hatékony, átlátható kezelésére. Rendkívül flexibilis több fajta adatbázis rendszert támogat, és amennyiben a program struktúrája megfelelően van kialakítva szinte szó szerint 2 kattintással kilehet cserélni a program mögött lévő adatbázist.

Eleinte a mysqli függvényeit használtam aztán inkább később mint hamar átálltam a pdo-ra, a projektemben nem sok gyakorlati előrelépést (kevesebbet kell az escapeléssel foglalkozni a prepared statementek is hasznosak) hozott azon kívül hogy már találkoztam ezzel a technológiával és használtam, elmondhatom hogy a legmodernebb technológiákat használom. Jelenleg 2 kis osztályt használok hozzá:



6. ábra: adatbáziskapcsolat

Ebben az osztályban definiálom a jelenlegi adatbázis kapcsolat. A fejlesztő annyi ilyen osztályt hoz létre optimális esetben amennyi adatbázissal akar dolgozni. Jelen példához a változók elnevezése nem a legjobb.



7. ábra: Kapcsolat

Ezen az osztályon keresztül létesítek kapcsolatot az adatbázissal. Mindenegyes funkcióban ahol kapcsolatot létesítek példányosítom és ezáltal a konstruktor beolvassa az előző osztályban lévő értékeket, így hogyha azt az osztályt módosítom vagy itt az elérési utat akkor egy egészen más adatbázissal is működne (elméletileg a query-k nem minden helyen működnek ugyanazzal a syntaxal).

## Biztonság

„A biztonságot befolyásoló tényezőket két csoportra lehetne osztani. Egyrészt függhetnek a szerver beállításaitól, amelyek főleg a rendszergazdát érintik, másrészt függhetnek maguktól a futtatott PHP programoktól, amelyekért már a programozó a felelős. Valamennyire szétválasztható a kettő egymástól, de hiába ír a programozó majdnem tökéletes kódot, ha egyetlen kis hibát kihasználva hozzá lehet férni nem publikus tartalmakhoz is, mert nincs az Apache, illetve a PHP beállításai között korlátozva a fájlok elérhetősége. Ez persze fordítva is igaz, érdemes összhangban tartani a kettőt.” -http://weblabor.hu/cikkek/phpbiztonsag

Jelen helyzetben mind a programozó és a rendszergazda szerepe is rám hárult, de mivel ez egy programozói szakdolgozat, ezért a rendszergazdai biztonsági intézkedések a háttérbe szorultak.

## Felhasználók azonosítása és kezelése

A felhasználók azonosítására a beléptetésénél szükségünk lesz. A felhasználóknak azonosítaniuk kell majd magukat, hogy jogosultságot szerezzenek bizonyos funkciók eléréséhez. Az autentikáció során két adat megadására kötelezzük őket: a email és a hozzá tartozó jelszót kell majd megadniuk - amit összevetünk az adatbázisban tároltakkal. Mindezek azonban bizonyos biztonsági lépéseket is megkövetelnek, a külső támadások kivédése miatt.

## A jelszavak titkosítása

A jelszókat a következőképp tároljuk: a karakterlánchoz hozzáadunk egy másik karakterláncot, amit egyirányú kódolással elmentünk az adatbázisba. Ez az egyirányú kódolás olyan kódszót képez, amiből nem tudjuk közvetlenül az eredeti információt visszanyerni. Így beléptetéskor a felhasználó által megadott jelszóból ugyanezzel az eljárással mindig kódszót kell képezni és összevetni azt az eltárolttal. Ebből következik, hogy a jelszóhoz hozzáadott karakterláncot is el kell majd tárolnunk,hiszen az összevetés során szükségünk lesz.



8. ábra: titkosított jelszó tárolása

## A felhasználók beléptetése

A HTTP protokoll nem képes kéréseket egyedi felhasználókhoz kötni, ezért munkamenet-kezeléssel oldottam meg beléptetést, hogy ne kelljen minden oldalkéréskor jogosultságaikat megadniuk. Az autentikáció tehát megköveteli egy beléptető felület létrehozását, ahol azonosíthatják magukat a felhasználók. Sikeres azonosítás után a munkamenetben elhelyezzük és minden válasszal elküldjük a felhasználói azonosítójukat, amit minden kéréskor ellenőriznünk kell. Itt bevezetünk egy újabb biztonsági intézkedést, a munkamenetek csak adott időintervallumig lesznek érvényesek - például 20 percenként lejárnak.

## Tesztelés

A programkódok tesztelését mindig kézzel végeztem, úgy éreztem még nem eléggé összetett az oldalam hogy egységteszteket írjak hozzá. Néhány helyen bent maradtak olyan kódok amik a tesztelést teszik könnyebbé de mivel az oldalat még tovább fogom fejleszteni ezért úgy gondoltam bent hagyom még őket.

## Osztályok, mappák tartalmának részletesebb elemzése

**Checkers:** A checkers nevezetű mappában tesztek vannak amik általában formok leellenőrzésére vannak megírva. Minden egyes PHP fileban egy olyan osztály van ami egy absztrakt osztályból öröklődik, ez azért van hogy mindenképpen benne legyenek a szükséges funkciók (jelen esetben csak 1 van). Ezeket a teszteket dinamikusan hívja meg másik osztály egy tömb alapján.

**Classes:** Itt néhány kisseb osztály van amik másik mappába logikailag nem illetek volna bele úgyhogy itt kaptak helyet.

* **Register:** Ez az osztály kezeli egy regisztációs form kitöltése az adatokat, kielemzi, ha hibát talál, azokat kiírja a felhasználónak ha pedig nem talál akkor az adatbázisba elmenti a felhasználó adatait és továbbítja egy belépési oldalra.
* **Signin:** Ez az osztály is egy form kitöltése után kezeli az adatokat, jelen esetben a belépéset.
* **Isloggedin:** Ez az osztály egy statikus funkciót tartalmaz ami azt ellenőrzi hogy a felhasználó belépett-e, ha igen nem történik semmi, ha pedig nincsen, akkor átirányítja a főoldalra. Ennek az osztálynak a tartalma minden egyes funkció elején meghívódik ahol szükséges hogy a felhasználó belegyen jelentkezve.
* **Values:** Ebben az osztályban statikus függvények vannak amelyekben olyan adatok tárolódnak amik nagyon ritkán változnak meg és nem lenne érdemes őket adatbázisban tárolni. Sajnos a PHP licitációi miatt nem lehet őket olyan tömbökben tárolni amelyeknek a tartalmuk megváltoztathatatlan (constant) így nem annyira biztonságos a használatuk de jelen esetben ettől eltekinthetünk.
* **AvailabilityCalc:** Ebben az osztályban egy 3 soros statikus függvény található ami egy hirdetés érvényességét számolja ki 2 adatból.

**Connections:** Ebben a mappában 2 olyan file található amik az adatbázissal való kommunikálást segítik.

**Controllers:** Ez az MVC-ben a C betű. Ebben a mappában találhatóak azok az osztályok amik az adatbázist összekötök a megjelenítéssel, itt történik a buisniess logic egy része is.

* **AdController:** Ez kezeli a hirdetéssel kapcsolatok oldalakat, ellenőrzéseket. Ezen keresztül érhető el a hirdetésfeltöltés funckció, ezzel lehet kilistázni a saját hirdetéseimet. Ez az egyik legnagyobb osztályom a 300 sorával, igyekeztem 150 sor körül tartani az osztályok méretét az átláthatóság miatt, úgyhogy ezt már érdemes lenne kisseb osztályokra bontani.
* **CommentController:** Ez az osztály kezeli a kommenteket a hirdetések alatt, ezen keresztül lehet feltölteni, ezzel van megoldva hogy minden kommentnek legyen külön URl-je. Főleg az AdController használja ennek az osztálynak a funkciót.
* **CronController:** Kisebb osztály egyetlen egy funkció van benne amit a szervernek bizonyos időnként lekell futtatnia. Ez a funkció küldi ki az üzeneteket és állítja be az értékeléseket egy licit megnyerése után. Minél gyakrabban van lefuttatva annál jobb, de túlzásba sem kell esni nehogy túl sok erőforrást felhasználjon.
* **IndexController:** Ez az osztály kezeli a főoldalat, a keresés funkciót, és a felhasználói profiloknak megjelenítésének egy részét.
* **MessageController:** A felhasználók és az oldal ezzel/ezen a funkción keresztül küldhetnek üzeneteket. Ez az osztály felelős a felhasználó üzenetek ki listázásáért, megtekintésért is.
* **NavbarController:** Ez az osztály generálja ki az oldalon lévő navigációhoz szükséges gombokat.
* **PaginationController:** Ez a lapozó, ez az osztály jeleníti meg a keresés után a lapozót az oldal alján.
* **RateController:** Ez az osztály felelős a értékelés megjelenéséért a sikeres leütés, aukció után a termék oldalán, az adatbázisban az ehhez tartozó rekord létrehozásáért.
* **RegisterController:** Ebben az osztályban vannak a regisztrációval kapcsolatos oldalak megjelenítéséért felelős funkciók, az ellenőrzéseket és az adatbázissal való kapcsolódást egy másik osztály végzi.
* **SalesController:** A vásárlással kapcsolat funkciók ebben vannak a leütésnél ez küldi ki az üzenteket a felhasználóknak, ez hozza létre az értékeléseket.
* **SigninController:** Ez felelős a bejelentkező oldal megjelenítéséért, valamint a kijelentkezésért is.
* **SpecSearchController:** Ebben az osztályban lettek volna a speciális kereséssel kapcsolatos funkciók de ez sajnos nem lett befejezve.
* **UserController:** Ebben van a felhasználó adásvételeiből való értékelések kiszámítása, ezek az adatok is megjelennek a felhasználó profilján, átlagértékelés, összes adásvétel…

**View:** Ez az MVC V betűje ebben vannak a templatek amiket az előbbiekben részletezett controllerek felhasználnak a megjelenítéshez. Ezekből a templatek-ből túl sok van hogy mindegyiken végigmenjünk, igyekeztem használatukra utalóan elnevezni őket.

**Css, fonts:** Itt vannak a megjelenítéshez használatos css-ek és betűtípusok, a css-ek közül csak a Sajat.css írtam meg a többi pluginekehez, bootstaphoz tartozik.

**Img, Upload:** ebben a két mappában vannak eltárolva a képek, az img mappában olyan képek vannak amik az oldal működéséhez szükségesek, az upload mappában pedig a felhasználók által feltöltött képek vannak

**JS,lib,plugins:** Ezekeben a mappákban külső forrásokból származó pluginok, könyvtárak és hozzájuk tartozó file-ok vannak.

**tmp:** A framework által dinamikusan kigenerált filek amiket be cache-el és ezáltal gyorsabb az alkalmazás.

# Felhasználói dokumentáció

## Oldal felépítése

Minden oldal 3 fő részből áll:

* Navbar
* Content
* Footer

Amikor egy oldal tartalmáról fogok beszélni a későbbiekben mindig a contentről lesz szó.

**Navbar:** Ez egy dinamikusan kigenerált felület amivel a felhasználó az oldalon navigálhat.

**Content:** Ez az oldal tartalma, általában dinamikus, itt történik az adatok átadása a szerver-felhasználó között.

**Footer:** Statikus felület itt egyszerű adatok jelennek meg az oldalról: kapcsolat, hírlevél stb….

## Főoldal

Ez az oldal fogadja a felhasználót amennyiben fellép az weblapra. Itt van egy slider ahova lehet hirdetéseket rakni vagy bármilyen tartalmat, jelenleg üres. Ezen kívül az items táblából legújabb hirdetések egy részét kilistázza a főoldalra oszlopokba rendezve.

## Regisztrációs oldal

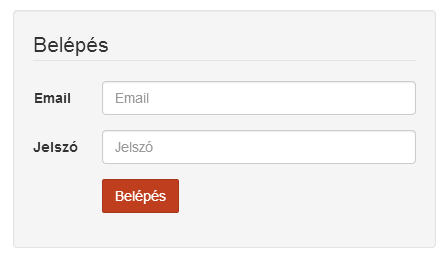
A legszükségesebb adatokat kell itt megadni:

* Email
* Felhasználónév
* Jelszó

Ezeket később bővítheti, pl bemutatkozással ha itt regisztrál minden felhasználói funkciót igénybe vehet.

## Belépés

Egy helyes email-jelszó párossal léphet be a felhasználó.



9. ábra Belépés

## Üzenetek

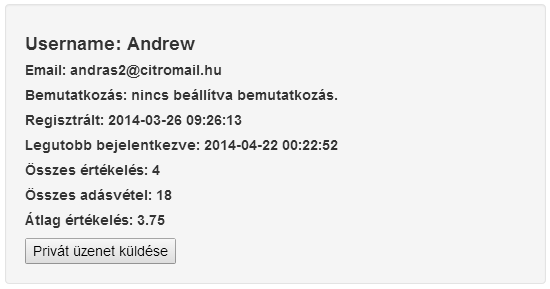
Itt a jelennek meg az üzenetek, értesítések, amelyeket a felhasználó kap a többi felhasználótól, az oldaltól. Ez az oldal csak akkor elérhető el hogyha a felhasználó bevan jelentkezve.

## Keresés

Ezzel egy adatbázisban hirdetések címében lehet keresni 50esével, írja ki a találatokat.

## Profil

Minden felhasználónak van egy profilja ami minden más felhasználó számára publikus, kivéve az elérhetőségek, amik csak akkor érhetőek el ha a két felhasználó már kereskedett egymással.



10. ábra: Felhasználói profil

## Privát Üzenet

Mint a neve ˙is mondja a felhasználok küldhetnek egymásnak privát üzeneteket. Akinek a szeretnénk üzenet írni annak profiljára kell menni és ott rákattintani a „Privát üzenet küldése” gombra.

## Kommentelés

A felhasználok a hirdetések alatt tudnak egymással kommunikálni, kérdezni az eladótól publikusan kommenteken keresztül. Ez egy nagyon hasznos funkció hogyha valaki kifelejt valami fontos adatot a hirdetéséből, azt csak egyszer kell megkérdezni, és az összes többi felhasználó azonnal látja és nem kell mindenkinek üzenetben megkérdeznie ugyanazt. Minden egyes komment rendelkezik saját url-el. Ezáltal a felhasználók könnyen hivatkozhatnak rájuk.

## Hirdetés feladása

Itt tudnak a felhasználók feladni hirdetéseket mindenféle paraméterrel. A hirdetéseket 3 fő részre lehet osztani:

* **Fixáras:** A fixáras aukcióknál a hirdető megad egy árat, amin bármikor meglehet vásárolni. (lejárati idő előtt)
* **Licites:** Feltöltéskor a felhasználó megad egy minimál árat valamit egy értéket a licitlépcsőnek. Amikor valaki megakarja vásárolni a hirdetett árut, licitálnia kell rá, a licitnek mindig minimum a licitlépcső értékével kell növekedni. Az a vásárló fogja megkapni a terméket akinek akkor van a legnagyobb érvényes licité a tárgyon amikor az lejár.
* **Licites & fixáras:** Ennél a fajta hirdetésnél ha valaki leüti a terméket azonnal megveheti és érvénytelenné válik a hirdetés, ha pedig senki nem üti le és a lejártakor van licitáló akkor az övé.

## Hirdetéseim

A felhasználó által feladott összes hirdetést listázza ki.

## Hirdetés-oldal

Ez egy dinamikusan kigenerált oldal ahol a felhasználók egy bizonyos hirdetés adatait nézhetik meg, kommentezhetnek. Itt a termék képe, ára, leírás, hirdető neve és mindenféle hasznos adat jelenik meg.

# Github

A github egy verziókövető rendszer amely szoftverfejlesztést segíti és a nyílt forráskódú projektek megosztását. Több nagy projekt is ezt a rendszert használja pl: Android, Node.js, JQuery. A github főbb funkciói:

* Verziókezelő
* Issue tracking
* Wiki
* Közösségi tér
* Forks

**Verziókezelő:** Verziókezelésre a Git rendszert használja, ennek az a lényege hogy a felhasználóknak van egy „repository”-juk a gépükön amiben a projektjüket tárolják és van egy közös tárhely a szerveren. A megfelelő jogosultsággal rendelkező fejlesztők mindig feltöltik a saját frissítéseket a közös tárhelyre a többiek pedig letöltik. Amikor valami-t változtatnak a forráskódon (saját gépükön) és szeretnék azt publikálni akkor leellenőrzik hogy meggyezik-e a verzió a szerveren lévővel ha igen akkor egy kis magyarázattal feltöltik a változtatásukat, ezt nevezik „commit”-nak. Amikor nem egyezik a saját verziójuk a szerveren lévővel akkor előbb le kell tölteni a legújabbat és „merge”-elni (összeolvasztani) a sajátjukkal, és csak ezután tudják feltölteni. A szerver minden egyes commitot eltárol, különbségeket a verziók között ez azért hasznos mivel megkönnyíti a code-reviewet (a jó fejlesztők mindig leellenőrzik egymás kódját hogy nehogy rosszminőségi kód kerüljön a projektbe), ha valami olyan frissítés kerül feltöltésre ami ’nemkívánatos’ akkor egész egyszerűen visszalehet állítani egy régebbi verziót.

**Issue tracking:** Ez egy hibakezelő rendszer a lényege hogy a projektnek az oldalán hozzá lehet adni „ticketeket” amik hibákat vagy megoldandó feladatokat jelölnek, ezekhez lehet komenneteket fűzni stb.. Nagyon hasznos feature amennyiben a fejlesztő agilisan fejleszt, kijelöl egy adag ticketet hogy ezeket akarja megcsinálni a heti/havi sprintjében. Nagyon jól szemlélteti a hátralévő feladatokat.

**Wiki:** Ezzel a felhasználó a projektjéhez egy kis wiki oldalat adhat hozzá, amelyben részletesebben ismertetheti a funkciókat, a telepítéshez instrukciókat adhat stb…

**Közösségi tér:** A github nagyon hasonlít a facebookhoz a felhasználóknak a főoldalon ugyanúgy megjelenik egy „news feed” a vele kapcsolatok projektekről, fejlesztőkről. Lehet követni a többi felhasználót, híres fejlesztőket, megnézni hogy miket írnak.

**Forks:** Az úgynevezett forkok egy-egy projekt saját célú másolatai amiket mi magunk fejleszthetünk tovább. Nagyon hasznos hogyha megtetszik egy projekt de nem tetszik milyen irányba fejlődik, ekkor készíthetünk egy forkot, amelyben az egész projekt lemásolódik a saját profilunkhoz és ezután saját ízlésünk szerint fejleszthetjük tovább. A népszerűbb projekteknek gyakran van több száz forkjuk.

Én is elkezdtem használni a piactér fejlesztése közben, nagyon hasznosnak találtam még úgyis hogy csak egyedül fejlesztettem, jó látni visszamenőleg hogy az ember mikor mit csinált. A kedvenc funkcióm az issue tracking lett, amit sajnos későn fedeztem fel ezért csak a végső fázisban tudtam használni. Ezeken a funkciókon felül nagyon hasznos, hogy egy szép grafikus felületen (mindenféle színes grafikon, átlátható felület) foglalja össze a projektet ezért könnyen bemutatható referenciaként, vagy akár önéletrajzba is beírható hogy ezen és ezen a projekten dolgoztam.

11. ábra: A github kabalafigurái.

# Összefoglalás

Úgy érzem, hogy a Piactér szakdolgozatnak egy jó választás volt, sikerült fejlődnöm, egy olyan oldalt készítettem amit szívesen mutathatok meg az ismerőseimnek. Egy aukciós ház implementálása szinte minden területet lefed - az adatbázis-struktúra valamint a felhasználói-felület megtervezésén át, a felhasználók és a dinamikus tartalmak kiszolgálásáig. Az alkalmazás megtervezése és elkészítése közben összefoglalhattam az eddig tanultakat, ráadásul időközben a PHP-nak és a Fatfree keretrendszernek köszönhetően új ismereteket is elsajátíthattam. Az elmúlt fél évben rengeteg tapasztalattal lettem gazdagabb, úgy érzem komoly szakmai előrelépést értem el a szakdolgozat elkészítésével.

# Telepítés és futtatás

A oldal beüzemeléséhez a szerverre egy EasyPHP vagy Wamp csomag kell ami tartalmaz:

* Apache szervert
* PHP értelmezőt
* MySQL adatbázist

Az adathordozón-n lévő piactér mappa tartalmát be kell másolni oda ahonnan a webserver host-ol. Létre kell hozni egy adatbáziskezelő szoftverrel (pl:navicat) egy adatbázist a mysqlben, ezen az adatbázison lefutattani az sqldump-ot. A piactér mappában lévő connection/pdoconfig.php-ban pedig beállítani az adatbázis kapcsolatot. Ezután a szerveren az ’oldal neve/cronjob1’ oldalt valamilyen időzített funkcióval megkell hívni minimum 5 percenként (minél gyakrabban annál jobb). Ezek után ahonnan elérhető a szerver, az oldal teljes funkcionalitással működik.

# Irodalomjegyzék

* Tanuljuk meg a PHP5 használatát 24 óra alatt - Matt Zandstra
* stackoverflow.com
* caniuse.com
* Tiszta kód - Az agilis szoftverfejlesztés kézikönyve - Robert C. Martin
* Túlélőkönyv programozóknak - Robert C. Martin
  + PHP
* http://www.php.net
* http://hu.wikipedia.org/wiki/PHP
  + MySQL
* http://hu.wikipedia.org/wiki/MySQL
  + Apache
* http://hu.wikipedia.org/wiki/Apache\_HTTP\_Server