SZAKDOLGOZAT

DEBRECEN
2006. április

DEBRECENI EGYETEM
INFORMATIKA KAR
INFORMATIKAI RENDSZEREK ÉS HÁLÓZATOK
TANSZÉK

WEBES ALKALMAZÁSFEJLESZTÉS

EGY WEBÁRUHÁZ LÉTREHOZÁSA

Témavezető: Dr. Kuki Attila
Egyetemi adjunktus

Készítette: Óvári Csaba
Programozó matematikus
Esti tagozat

DEBRECEN
2006. április

Tartalomjegyzék

1. Bevezetés

1.1. Bevezető

Az 1990-es évek elején az informatika, számítástechnika óriási fejlődésen esett át. A fejlődés robbanásszerűen ment végbe, különösen a hardver tekintetében. A számítógépek egyre gyorsabbak lettek, szinte évről évre megduplázódott a processzorok sebessége, nőtt a memória kapacitás, illetve a háttértárolók mérete. Ezzel párhuzamosan fejlődésnek indultak az informatikai hálózatok, adatátviteli eszközök, ugrásszerűen megnőtt a hálózatok adatátviteli sebessége, különböző adatátviteli protokollok alakultak ki. Ezek hatására az ember felismerte, hogy a számítógép most már nem csak munkaeszköz lehet, hanem szórakozást, hatékony kommunikációt jelent és nem utolsó sorban, az internet térhódításával kitárja a világot előttünk.

Az internet előde az ARPANET volt, melyet az Egyesült Államok védelmi minisztériumának utasítására hoztak létre. Létrehozásának oka, hogy szükség volt egy olyan kommunikációs rendszerre, amely gyors és biztonságos információcserét valamint állandó rendelkezésre állást biztosít. Ekkor még persze nem is gondolták, hogy ezzel egy világméretű bárki által használható, elektronikus hálózatot hoznak létre. Az internet előbb-utóbb kilépett a csak katonai ill. tudományos felhasználók köréből és bárki számára elérhetővé vált. A felhasználók összetételének kiszélesedésével ill. számuk rohamos növekedésével az igények is megváltoztak. Megjelentek az internet új felhasználási formái, kialakultak az első levelezőrendszerek, a telnet, e-mail, megjelentek az első weboldalak.

Már az 1970-es, 80-as években léteztek hálózatos környezetben futó alkalmazások, azonban ezek a hálózatok még kezdetlegesek, lokálisak voltak, így maguk a szoftverek sem léptek ki ebből a körből. Kialakultak az első hálózati protokollok, melyek később az internet alapját is képezik. Ezek a protokollok, már nem csak egy helyi hálózaton tudtak kommunikációt biztosítani, hanem egy világméretűn is. Tehát már készíthetők olyan alkalmazások, melyek a világ minden pontján elérhetőek, használhatóak. Vagyis kialakul az internetes alkalmazásfejlesztés.

1.2. Témaválasztásom

Az internet térhódításával, a legkülönfélébb felhasználói igények merültek föl. A felhasználók először megelégedtek a különféle levelezőrendszerekkel, on-line és egyben interaktív csevegő szoftverekkel, fórumok olvasgatásával, egyszerű internetes játékokkal, illetve a honlapok puszta böngészésével. Később alakultak ki az első igazán hasznos webes alkalmazások, mint például a webáruházak, ami on-line vásárlást tesz lehetővé, a különböző hivatali, banki ügyintézést lehetővé tevő honlapok, az elektronikus távoktatást megvalósító site-ok és sorolhatnám a végtelenségig. Manapság tulajdonképpen minden, amit az emberi képzelőerő el tud képzelni az interneten megtalálható. Az internet, én saját magamon is tapasztalom, hogy az ember életének részévé válik.

Mintegy három-négy évvel ezelőtt kezdtem „felfogni” az internet jelentőségét és kezdtem el foglalkozni vele. Először én is csak böngészgettem, chat-eltem, fórumokat olvasgattam, utána viszont elkezdett érdekelni, hogy valójában mi is van a honlapok mögött. Vagyis, hogyan jeleníthetjük meg, hogy lesz egy statikus weboldalból dinamikus, mi is az a webszerver, milyen nyelven íródnak, milyen adatbázisszervert használnak és ezeket hol is találom, ill. hol helyezhetőek el. Ezen kérdéseket csak úgy tudtam megválaszolni, ha beleástam magam a szakirodalomba, érdeklődtem az internetes fórumokon, ismerőseimtől kértem ötleteket, tanácsokat. Hamar körvonalazódott előttem, hogy mire is van szükségem egy dinamikus honlap megszerkesztéséhez. Kellett egy fejlesztőkörnyezet, egy programnyelv, egy webszerver illetve egy adatbázis-motor.

Ezek kiválasztásánál a fő szempontjaim a következők voltak:

- legális szoftver
- ingyenes legyen
- később is használható legyen
- széles körben elterjedt
- bőséges dokumentációval rendelkezzen
- és nem utolsó sorban elérhető legyen

Ezen szempontok után már nem sok „jelentkező” maradt, így esett a választásom a PHP, MySQL, Apache hármasra. Ezen eszközök segítségével már egy jó ideje fejlesztgetek magamnak kisebb alkalmazásokat, mint például társkereső oldalt ill. használtautó keres-kínál oldalt. Persze ezeket önszorgalomból készítettem és nem üzleti alkalmazásként. Szakdolgozatom témáját ezek függvényében akartam megválasztani, tehát egy, a már megszerzett ismereteimre épülő, olyan alkalmazást szerettem volna létrehozni, ami kihasználja a PHP ill. MySQL lehetőségeinek nagy részét és megfelelő színvonalú egy szakdolgozathoz. Némi gondolkodás után határoztam el, hogy egy webáruházat fogok készíteni, mégpedig egy dvd filmeket árusító áruházat. Szakdolgozatom célja, hogy a talán legelterjedtebb internetes fejlesztőeszközök segítségével, bemutassam egy webes alkalmazás létrehozását egészen az alapoktól.

2. Webes alkalmazásfejlesztés

2.1. Statikus weblapok

Alapvetően statikus weblapoknak nevezzük az olyan HTML dokumentumokat tartalmazó oldalakat, amelyek állandó tartalommal rendelkeznek. Tehát ezek nem interaktív oldalak, csupán olvasni lehet őket. Tulajdonképpen csak információk közlésére alkalmasak, semmi többre. Manapság egyre kevesebb az ilyen oldal, inkább csak a szoftverfejlesztés iránt érdeklődők, illetve tanulók készítenek ilyet.

2.2. Dinamikus weblapok

A dinamikus weblapok interaktívak és tartalmuk folyamatosan változik, a felhasználó személy tevékenykedésétől függően. Dinamikus weboldalak esetén szükség van webszerver alkalmazására, melynek segítségével tudjuk, a böngésző által küldött információkat feldolgozni és a választ, vagyis a kimenetet visszaküldeni a böngészőnek. Ilyen szerver-oldali alkalmazások közé sorolhatók a különböző CGI programok, melyek különböző nyelveken íródhatnak, de talán ezek egyre inkább kezdenek háttérbe szorulni és inkább a PHP, Java, JSP, ASP nyelvek használata válik „kötelezővé”. Dinamikus weblapok alkalmazásakor, igen gyakran használunk valamilyen adatbázist is, az adatok tárolására.

2.3. Alkalmazásom fejlesztőeszközei

Először a PHP-ről szeretnék egy pár mondatot megemlíteni. A PHP elődje a PHP/FI volt, amelyet Rasmus Lerdorf készített 1995-ben. Kezdetben egy egyszerű Perl szkriptgyűjtemény volt, majd később ő egészítette ki egy terjedelmesebb C nyelvű implementációval, amely már képes volt adatbázisokkal kommunikálni, és lehetővé tette, hogy a felhasználók egyszerű dinamikus webes alkalmazásokat fejlesszenek. A mai PHP elődje a PHP 3 volt, ami 1997-ben jelent meg egy scriptnyelvként. Előnye, hogy ingyenes és könnyen elsajátítható, azonban eszközrendszere igencsak szegényes volt.

A PHP-t weboldalak szerver-oldali programozására tervezték, illetve több adatbázisrendszer elérését tették lehetővé. A nyelvet később továbbfejlesztették, most már támogatja az objektumorientált paradigmát, illetve új függvénykönyvtárakkal is fel van vértezve. Jelenleg a PHP 5-ös verziónál tartunk, ami hivatalosan 2004 júliusában jelent meg. Szakdolgozatomhoz a PHP-4.3.4-Win32-es verzióját használtam.

Egy kis MySQL. A MySQL-t eredetileg is webalkalmazások létrehozására tervezték, ahol az adatbázisoktól elvárt legfontosabb tulajdonságok a gyorsaság, méretezhetőség és az egyszerű felügyelhetőség. E célok elérése miatt néhány alapvető relációs adatbázis tulajdonság a háttérbe szorult, különösen a régebbi verziókban. Azonban a MySQL 4.0 és 4.1-es verziójától, már az adatbázisrendszer részét képezik mind a tranzakció-kezelés, mind az idegen kulcsok ill. az egymásba ágyazott lekérdezések használatának lehetősége. A MySQL a világon a legszélesebb körben használt nyílt forráskódú adatbázis-kezelő, több millió felhasználóval, mind egyéni felhasználó, mind vállalati szinten. Az alkalmazásfejlesztők előszeretettel használják, mert nagyon gyors és messze nem olyan összetett, mint például az Oracle. A MySQL is egy többszálú végrehajtást támogató kiszolgáló, tehát minden egyes új kapcsolat esetén, egy új kiszolgáló folyamat indul el.

Legfőbb jellemzői:

- gyorsaság
- hordozhatóság
- bármely más nyelven írt felület illeszthetősége
- kedvező ár, illetve ingyenes

Szinte minden nyelv alkalmas arra, mint például PHP, C, C++, PERL, JAVA, hogy felhasználói felületet írjanak hozzá. Jelenleg a MySQL 5.0-s verziónál járunk, én a dolgozatomhoz a 4.0.4-es verziót használtam.

És végül az Apache. Az interneten több fórumon is érdeklődtem, illetve néztem a szakmai oldalakat és mindenhol az Apache-ot, mint webszervert ajánlották amennyiben PHP-ban akarok programozni. Elfogadtam az okosabbak tanácsát, természetesen egyáltalán nem bántam meg. Bőséges dokumentáció áll rendelkezésre a telepítéséhez, illetve a kapcsolat megteremtéséhez a PHP-val. Jellemzői továbbá, hogy stabil, biztonságos, rugalmas, jól konfigurálható és természetesen ingyenes. Alkalmazásomhoz az Apache 2.0.43-as verzióját választottam.

2.4. Internetes biztonság

A megfelelő internetes biztonság elérésével, jelenleg a kriptográfia, vagyis a titkosítástudomány foglalkozik. Az internetes kereskedelem során az utóbbi hét évtizedben kifejlesztett rejtjelezési algoritmusokat használják. Az internetes biztonságot leginkább egy lánchoz lehet hasonlítani, mivel annak erősségét is a leggyengébb láncszem határozza meg. Hiába vannak megbízható, kellően bonyolult matematikai algoritmusokat felhasználó biztonságos kommunikációt lehetővé tevő rendszereink, ha a rendszer hardveres, vagy szoftveres megvalósítása nem megfelelő.

Napjaink titkosítási algoritmusai jelenleg már megfelelő védelmet nyújtanak az adatok illetéktelen felhasználása ellen, ez azonban sajnos nem jelenti azt, hogy az ezeket használó rendszerek is ugyanilyen biztonságosak lennének. Az internet

térhódítása folyamán mind több részről fogalmazódtak meg komoly kétségek az internetes kommunikáció biztonságával kapcsolatban. Az internetes kereskedelem robbanásszerű bővülésével, az illetéktelen információszerzéssel kapcsolatos visszaélések száma és visszaélések értéke is növekedetett.

Ahhoz, hogy egy dokumentumot, egy fájlt vagy egy interaktív kommunikációt biztonságban tudhassunk, a következők szükségesek:

- **authentikáció -** személyek azonosításának biztosítása
- **authorizáció** - hozzáférés korlátozása
- **titkosság, bizalmasság**
- **adatintegritás** - adatsértetlenség biztosítása
- **letagadhatatlanság -** egy objektum létrehozója ne tagadhassa le az objektum létrehozásának tényét.

A titkosítás célja:

A modern kriptográfia, azaz titkosítástudomány, a matematikai eszközök segítségével több célt is szolgál. A legfontosabb és legnyilvánvalóbb feladata, hogy egy adott bizalmas, vagy nagy értékű információ ne juthasson illetéktelenek birtokába. Más szóval a titkosítás egyik kiemelkedően fontos területe, hogy két, vagy több fél kommunikációja alatt, csak az arra jogosultak férhessenek hozzá, illetve változtathassák meg a titkosított információkat. Bár a fent említett feladat elvégzése a kriptográfia alapvető célja, a gyakorlati életben hagyományos feladatánál – adatok titkosítása, és visszafejtése – sokkal szélesebb körben hasznosíthatjuk szolgáltatásait.

A mai modern titkosítástudomány egyik széles felhasználási területe az internetes világ, azon belül az internetes kereskedelem, ahol a bizalmas adatok védelme mellett fontos lehet a kommunikáló felek azonosítása, illetve az egyszer elküldött megbízás letagadhatatlansága is. Bármely kommunikáció során egy üzenet egy feladótól, egy kommunikációs csatornán keresztül jut el a címzetthez. A titkosítás alapfeltétele ezen kívül, hogy a bizalmas kommunikációt egy harmadik fél (a támadó) megpróbálja lehallgatni.

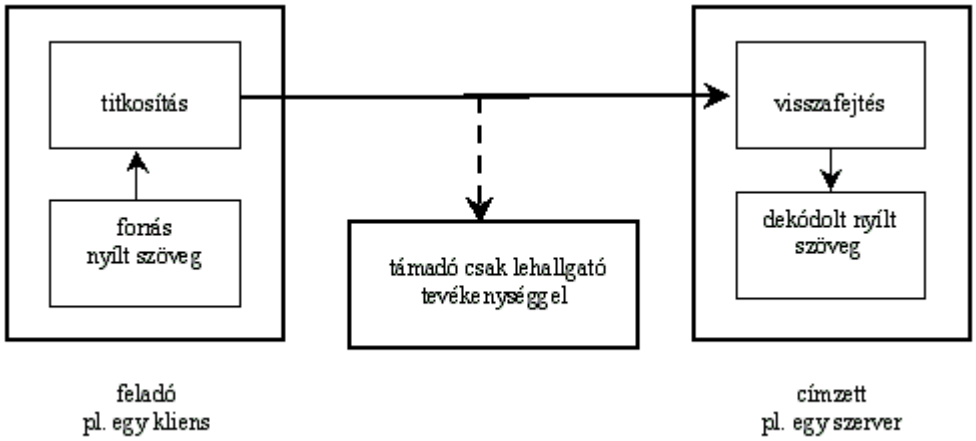
Közelebbről szemlélve egy támadónak két célja lehet:

- jogosulatlan hozzáférés az üzenet tartalmához
- hamis üzenet küldése a címzettnek

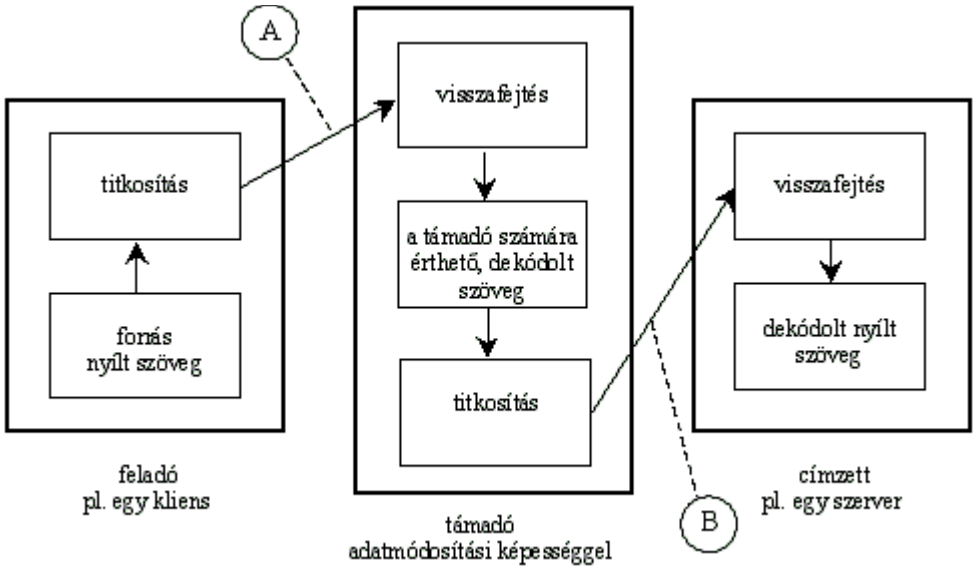
Egy kommunikációhoz való csatlakozás alapján a támadások két típusát különbözteti meg a szakirodalom:

- passzív
- aktív

Passzív támadásról beszélünk, ha a támadó hozzá tud férni a bizalmas információkhoz, de nem tudja megváltoztatni a kommunikációt, azaz hamis információt nem tud indítani a címzettnek. A támadás célja az információszerzés.



Aktív támadáskor a támadó képes adatokat megváltoztatni és ki tudja magát adni a címzett, vagy a küldő félnek. Az ilyen típusú támadások okozhatnak a legnagyobb gazdasági kárt, - például egy bank esetében hamis tranzakció indítása.



A titkosító módszerek közös jellemzője, hogy egy kódolatlan (plaintext) információhalmazból olyan kódolt információ halmazt (chipertext) készít, melyet csak pótlólagos információ (kulcs) birtokában tudunk visszafejteni. Az előzőekből gondolom mindenki számára láthatóvá vált, hogy a kriptográfia nagyon fontos, hanem az egyik legfontosabb részévé vált a biztonságos internetezés világában. Az idő folyamán, szinte hihetetlen gyorsasággal az internetes biztonság vagyis az „e-kriptográfia", bizony egy külön tudományággá nőtte ki magát. Jelentősége, óriási súllyal bír az elektronikus kommunikációban.

2.5. Webdesign

Az internetes megjelenés az egyik leghatékonyabb hirdetési és tájékoztatási lehetőség, ahol korlátlan mennyiségű információt közölhetünk a célközönséggel és a nézelődőkkel egyaránt. Egy honlap tervezésénél lényeges kérdés a megjelenés formája. Nem mindegy, hogy a honlapunk tartalma milyen formában jelenik meg. Az érdeklődők általában az első, pillanatnyi benyomásuk alapján döntik el, hogy felvegyék-e a kapcsolatot a honlap tulajdonosával, avagy sem. Ezért rendkívül fontos a megfelelő design kialakítása. Most meg szeretnék adni néhány alaptételt, ami elvezethet egy sikeres honlap elkészítéséhez.

Lássuk:

- Hierarchia: A honlap legfontosabb gondolatai, elemei legyenek azonnal szembetűnők.
- Tisztaság: A design csak akkor lehet igazán sikeres, ha áttekinthető, vagyis magát az üzenetet nem pedig a közvetítő közeget állítja a középpontba.
- Tördelés: Segíti a kommunikációt, ha egy hosszú üzenetet –például alcímekkel, különböző méretű részekre tagolunk.
- Sebesség: Az üzenet közlése akkor jó, ha az olvasó egy szempillantás alatt felismeri a honlap célját, lényegét.
- Arculat: Egy honlap még hatásosabbá tehető, a megfelelő színek és betűtípusok alkalmazásával.

A design jelentősége:

- Figyelemfelkeltés: A jó design segít, hogy az információáradatban észrevehető legyen az üzenet. A designnak vonzania kell a tekintetet.
- Meggyőzés: A hatásos design meggyőzi az olvasót a közlendő f ontosságáról, illetve magabiztosságot, szakmai hozzáértést sugall.
- Olvashatóság: Jó designnal olvashatóvá tehetjük a közlendőket, - például tördelünk, kiemelünk, nem használunk apró betűket stb.
- Kiemelés: A jó designnak köszönhetően az információ világosan, meghatározott hierarchiába szervezve jelenik meg, ami megkönnyíti az olvasó számára, hogy elkülönítse a fontosat, a kevésbé fontostól.

2.6. Alkalmazástervezés általánosságban

Az alkalmazástervezés során a legnagyobb hibát akkor követhetjük el, ha nem vagyunk eléggé előrelátóak. Ezért a tervezés során először mindenképp alaposan át kell gondolnunk, hogy milyen alkalmazást is akarunk fejleszteni, tehát fel kell vázolnunk, hogy - például mit és milyen adatokat fogunk tárolni az adatbázisban, hogyan kapcsolódnak egymáshoz az adataink, méretezhető-e illetve skálázható-e a rendszerünk.

A tervezés lépései:

- célok meghatározása
- adatszerkezetek (táblák, mezők) definiálása
- adatok közötti kapcsolatok felismerése
- az üzleti logika illetve szabályok megfogalmazása
- végül az alkalmazás elkészítése.

Nagyon fontos, hogy az alkalmazás elkészítése az utolsó fejezet egy rendszer megvalósításában. Tehát először jól át kell rágnunk magunkat a feladaton, fejben ki kell tisztulnia, hogy mit is akarunk elkészíteni és csak azután fogjunk papírt és ceruzát. Kiindulhatunk nagy, tagolatlan táblákból, vázoljuk fel az összes adatot, amire szükségünk lesz. A következő lépés a normalizálás illetve a kapcsolatok felismerése. Adatbázisunkat a harmadik normálformáig mindenféleképpen bontsuk le, mert így egy átlátható, könnyen kezelhető, redundanciát kizáró táblarendszert kapunk.

Ha elkészült az előzetes adatmodell, akkor azt nézzük át az alkalmazás, vagy a leendő felhasználók szemszögéből is. Meg kell határoznunk az üzleti logikát, illetve szabályokat és meg kell néznünk, hogy az adatmodellünk megfelel-e ezeknek a szabályoknak. A tényleges kódolást csak akkor kezdhetjük el, ha az adatmodellt összhangba hoztuk az üzleti logikával és megfelel a támasztott követelményeknek.

3. A webáruház tervezésének folyamata

3.1. Követelmények meghatározása

Az alkalmazásunk egy DVD webáruház lesz. Mielőtt hozzá kezdenénk az adatbázis létrehozásához, illetve a kód megírásához, el kell döntenünk, hogy mit is kell tudnia az alkalmazásunknak. Tehát először rögzítenünk kell, az igényeket, az elvárásokat valamint fel kell állítanunk a rendszerrel szemben támasztott követelményeket.

Mivel egy webáruházat készítünk, így eleve következtetni tudunk annak céljaira, egyes részeire, elemeire, metodikájára, sőt még implementációjára is. A követelmények meghatározása során fel kell mérnünk a megrendelő igényeit, a leendő felhasználók elvárásait, figyelembe kell venni a szakmai szempontokat ill. a rendelkezésre álló időt és erőforrásokat.

Elvárásaink a webáruházzal szemben:

1. Regisztráció
   Először is lennie kell egy kitüntetett felhasználónak, nevezzük rendszergazdának.

2. Adatbázis karbantartás
   Tudnunk kell az adatbázisba új filmeket felvinni ill. törölni.

3. Böngészés
   A leendő vásárlók tudjanak böngészni a filmek között és ki is választhassák azt, amelyik tetszik nekik.

4. Kosár
   A vásárló bármikor megnézhesse, a kosarának tartalmát.

5. Módosítás
   A vásárlónak legyen lehetősége, a kosarának tartalmát módosítani.

6. Megrendelés
   A vásárlás véglegesítését jelenti, vagyis a vásárló megrendeli a kosarának tartalmát. Ekkor a megrendelt filmek raktári készlete csökken.

7. Tájékoztatás
   Miután a vásárló elküldte megrendelését, egy e-mailt kap, melyben tájékoztatjuk a vásárlás tényéről és ez egyben nyugtaként is szolgál neki.

8. Konzisztencia
   Biztosítani kell, hogy az ügyfél által kiválasztott filmek, tényleg bekerüljenek az ügyfél kosarába, valamint az ügyfél által a kosárból eltávolított filmek visszakerüljenek az adatbázisba. Megrendelés esetén a filmek raktári készlete csökkenjen, valamint az új filmek adatbázisba való felvitele és a régi vagy elfogyott filmek törlése zökkenőmentes legyen.

9. Adatbázis
   Biztosítani kell az állandó rendelkezésre állást, valamint, hogy az esetlegesen előforduló hibák esetén se történjen adatvesztés, az adatbázis mindvégig konzisztens maradjon. Biztonsági mentéseket kell készíteni.

Egyéb követelmények:

1. Használhatóság
   Az alkalmazásunk ne legyen túlbonyolítva, legyen egyszerű és funkcionális. A „mezei” felhasználók számára is átláthatónak és könnyen kezelhetőnek kell lennie.

2. Hibatűrés
   Vagyis a programnak fel kell készülve lennie, a „legelvetemültebb” hibák kezelésére is.

3. Hordozhatóság
   Az alkalmazásunk lehetőleg platform- és rendszer független legyen.

4. Validitás
   A programunk lehetőleg feleljen meg a HTML és egyéb szabványoknak. Különböző böngészőkben is ugyanúgy jelenjen meg weblapunk.

5. Adatvédelmi és biztonsági követelmények
   Programunk feleljen meg a biztonsági és adatvédelmi előírásoknak.

3.2. Az adatbázis elkészítése

Adatbázisomat a MySQL parancssoros ügyfélprogramjával, vagyis a MySQL monitorral hoztam létre. Ez a felület kiváló lehetőséget biztosít arra, hogy „kézzel” vagyis SQL utasítások használatával hozzuk létre a tábláinkat és ne valamilyen „varázsló” segítségével. Igyekeztem úgy létrehozni táblarendszeremet, hogy mellőzzön mindenféle redundanciát, illetve megfeleljen a harmadik normálforma követelményeinek. Így mintegy 13 táblára volt szükségem

Ezek a táblák a következők:

1. rendszergazda
2. film
3. felirat
4. hang
5. nyelv
6. foszerep
7. foszerep_kapcs
8. rendezo
9. kategoria
10. kosar
11. megrendelesek
12. megrend_tetelek
13. hirujsag

Az adatbázis tábláinak szerkezete, célja:

1. Rendszergazda tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
rendszergazda_id	smallint(5)	unsigned, primary key	not null	auto_increment
vezetek_nev	varchar(50)		not null	
kereszt_nev	varchar(50)		not null	
email	varchar(100)		not null	
felhasz_nev	varchar(30)		not null	
jelszo	varchar(20)		not null	

A rendszergazda táblában csak az arra jogosultak, tehát a rendszergazdák és az adatbázis adminisztrátorok vannak elhelyezve. Alkalmazásunkban nem tudunk új sorokat felvinni ebbe a táblába, illetve törlésre sincs lehetőség. Ezt közvetlenül SQL

utasítások segítségével tudjuk megtenni az adatbázis ügyfélprogramjainak egyikével. Mint látható a *rendszergazda_id* mező az elsődleges kulcs a táblában, értéke az *auto_increment* parancsnak köszönhetően, automatikusan növekszik. Értelemszerűen a rendszergazda a felhasználói nevével illetve jelszavával tudja elérni az adatbázist, az alkalmazáson keresztül és új filmeket vihet föl, illetve törölhet.

2. Film tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
film_id	smallint(5)	unsigned, primary key	not null	auto_increment
film_letre_dat	date		not null	
cim	varchar(100)		not null	
megjelen_ev	year(4)		not null	
jatek_ido	tinyint(3)	unsigned	not null	0
ar	smallint(5)	unsigned	not null	0
keszlet	smallint(5)	unsigned	not null	0
kategoria_id	smallint(5)	unsigned	not null	
rendezo_id	smallint(5)	unsigned	not null	

A film tábla tartalmazza a filmek adatait. Tehát a film címét, a film adatbázisba való bevitelének dátumát, a megjelenés évét, játék időt stb..

3. Felirat tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
felirat_id	smallint(5)	unsigned, primary key	not null	auto_increment
film_id	smallint(5)	unsigned	not null	
nyelv_id	smallint(5)	unsigned	not null	

Ebben a táblában tároljuk, hogy melyik filmhez milyen nyelvű felirat tartozik. Erre a táblára azért van szükség, hogy ne okozzon redundanciát az, hogy egy nyelvet több filmhez is hozzávegyünk, a film táblában. Tulajdonképpen egy kapcsolótábla, mely az elsődleges kulcson kívül csak két „külső kulcsot” tartalmaz. A MySQL általam használt verziója még nem ismeri a tényleges külső kulcs fogalmát.

4. Hang tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
hang_id	smallint(5)	unsigned, primary key	not null	auto_increment
film_id	smallint(5)	unsigned	not null	
nyelv_id	smallint(5)	unsigned	not null	

A tábla tulajdonképpen megegyezik a felirat táblával, csak itt az egyes filmekhez tartozó hangok /beszéd/ nyelvét tároljuk le. Szintén a redundancia elkerüléséért hoztam létre.

5. Nyelv tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
nyelv_id	smallint(5)	unsigned, primary key	not null	auto_increment
nyelv_megnevez	varchar(30)	unsigned	not null	

Ebben a táblában tároljuk le a nyelveket. Tehát itt soroljuk fel az összes általunk forgalmazott film nyelvét. Például: angol, francia, német, bolgár stb..

6. Foszerep tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
foszerep_id	smallint(5)	unsigned, primary key	not null	auto_increment
foszerep_nev	varchar(50)	unsigned	not null	

Itt a filmekben játszó főszereplőket adjuk meg, egyszerűen a főszereplők nevével. Alkalmazásunkban egy filmhez egy főszereplőt adhatunk meg.

7. Foszerep_kapcs tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
foszerep_kapcs_id	smallint(5)	unsigned, primary key	not null	auto_increment
film_id	smallint(5)	unsigned	not null	
foszerep_id	smallint(5)	unsigned	not null	

Ez is egy kapcsolótábla. A redundanciát hivatott csökkenteni. Ha –például a film táblában tároltuk volna le a film főszereplőjét, akkor valószínűleg lennének olyan esetek, mikor egy főszereplő több filmhez kapcsolódóan is tárolva lenne. Tehát a film táblában többször is megjelenne ugyanaz a név.

8. Rendezo tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
rendezo_id	smallint(5)	unsigned, primary key	not null	auto_increment
rendezo_nev	varchar(50)	unsigned	not null	

Itt a filmek rendezőinek neveit adjuk meg. Alkalmazásunkban egy filmhez egy rendezőt adhatunk meg.

9. Kategoria tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
kategoria_id	smallint(5)	unsigned, primary key	not null	auto_increment
kategoria_nev	varchar(50)	unsigned	not null	

Ebben a táblában a filmek kategóriáit adjuk meg. Mint például: vígjáték, akció, dokumentum, horror stb.. Ugyebár egy kategóriába akárhány film beletartozhat, ezért hoztam létre neki egy külön táblát.

10. Kosar tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
kosar_id	smallint(5)	unsigned, primary key	not null	auto_increment
munkam_id	varchar(32)		not null	
film_id	smallint(5)	unsigned	not null	
cim	varchar(100)		not null	
rendezo	varchar(50)		not null	
ar	smallint(6)		not null	0
termek_menny	smallint(5)	unsigned	not null	0
datum	datetime		not null	

Ez a tábla a „bevásárlókosár” implementációja. Tehát adatbázisunkban létre kellett hozni egy olyan táblát, ami az egyes felhasználók által kiválasztott filmeket tartalmazza. A leendő vevő a kosarába, szabadon rakhat, illetve ki is veheti a filmeket. A *munkam_id* mező a munkamenet azonosítót tartalmazza, ami nem lehet egyedi. Ha egyedivé tennénk ezt a mezőt, akkor a felhasználó egyszerre csak egy terméket tudna rendelni, ami nem igazán tenne jót az üzletnek. A következő mezők a kiválasztott film egyes alapadatait tartalmazza és a rendelt mennyiséget. Az utolsó mező a kosár utolsó módosításának idejét adja meg. Erre azért van szükség, mert sok felhasználó rak a kosarába termékeket, de végül soha nem jut el a tényleges vásárlásig. Ezek a félbeszakadt vásárlások bejegyzésre kerülnek ebbe a táblába és ott maradnak. De ezeket a tételeket így, a dátum alapján tudjuk törölni -például úgy, hogy töröljük a több, mint egy hete létrehozott kosarakat. Ezek a bejegyzések igen nagy valószínűséggel már nem lesznek megrendelve.

11. Megrendelesek tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
megrend_id	smallint(5)	unsigned, primary key	not null	auto_increment
datum	datetime		not null	
megrend_nev	varchar(50)		not null	
megrend_orszag	varchar(50)		not null	
megrend_irszam	varchar(10)		not null	
megrend_varos	varchar(50)		not null	
megrend_cim	varchar(200)		not null	
megrend_tel	varchar(25)		not null	
megrend_email	varchar(100)		not null	
termek_ossz	smallint(6)	unsigned	not null	0

A tábla a megrendelések adatait tárolja, mely egy kicsit le van egyszerűsítve, így ez esetben például feltételeztük, hogy a számlázási, vagyis megrendelési cím azonos a szállítási címmel. A táblában nem tároljuk a megrendelő hitelkártyájának adatait, ugyanis ehhez nagyon biztonságos rendszert kellene létrehoznunk. Tehát megfelelő titkosításra és egy webszervert védő tűzfalra lenne szükségünk.

12. Megrend_tetelek tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
megrend_tetelek_id	smallint(5)	unsigned, primary key	not null	auto_increment
megrend_id	smallint(5)	unsigned	not null	
film_id	smallint(5)	unsigned	not null	
megrend_menny	smallint(5)	unsigned	not null	0
ar	smallint(5)	unsigned	not null	0

Ebben a táblában tároljuk az egyes megrendelések tételeit. A *megrend_id* mező biztosít kapcsolatot a megrendelesek tábla megfelelő sorával. Érdekesség, hogy a film ára ebben a táblában is szerepel. Azt gondolhatnánk, hogy ez egy fölösleges ismétlődés, de gondoljuk csak át jobban a dolgot. A megrendelések összeállításánál hivatkozhatnánk a film tábla *ar* mezőjére, de ha egy megrendelés után megváltoztatjuk az árat -például csökkentjük, mert már kifutó film, akkor a megrendelés tételei is az aktuális tehát új árat tükröznék és nem azt, amin a vevő rendelt.

13. Hirujsag tábla

Mezőnév	Típus	Tulajdonságok	Null	Extra
hirujsag_id	smallint(5)	unsigned, primary key	not null	auto_increment
datum	datetime		not null	
nev	varchar(50)	unsigned	not null	
lakohely	varchar(50)	unsigned	not null	
eletkor	smallint(5)	unsigned	not null	0
email	varchar(100)			

És végül itt tároljuk le, a hírújságra jelentkezők alapadatait. Végül is, csak az e-mail cím a lényeges, amire megkapják az újságot.

4. Az alkalmazás létrehozása

4.1. Az alkalmazás szerkezete

Ebben a részben a webáruházat alkotó szkripteket, html-oldalakat szeretném bemutatni, és némi információt adni ezek implementációjáról. Ezen fájlok alkotják a rendszert, amelyek egy internetes webszerveren megtalálhatóak, természetesen működőképes formában.

Először nézzük meg a fájlok listáját:

File neve	Kiterjesztés	Méret
index	html	173 byte
kezdolap	html	904 byte
bal	html	818 byte
belepes	php	1306 byte
regisztracio	html	642 byte
naptar	php	2088 byte
kuldes	php	889 byte
bongeszes	php	3965 byte
ujfilm	php	8219 byte
filmtorles	php	2098 byte
kosarba_tesz	php	1151 byte
kosar	php	1908 byte
kosar_torles	php	436 byte
megrend_urlap	php	2470 byte
megrend_kuldes	php	3431 byte

És most lássuk, mi is ezen fájlok feladata:

1. index.html

Ennek a szkriptnek a feladata csak annyi, hogy két részre osztja a képernyőt és az egyik részbe betölti a bal.html oldalt, míg a másik, nagyobb részbe pedig a kezdolap.html oldalt.

```
<html>

<head>
<title></title>
</head>

<FRAMESET cols="20%,80%">
<FRAME NAME="bal" SRC="bal.html">
<FRAME NAME="jobb" SRC="kezdolap.html">
</FRAMESET>

</html>
```

2. kezdolap.html

Ez a kezdőoldala az alkalmazásnak. Ez a statikus lap, csak némi információt tartalmaz és egy-két képet. Funkciója tulajdonképpen nincs azon kívül, hogy ezzel a lappal találkozik először a kedves látogató.

3. bal.html

Ez az oldal a képernyő bal, vagyis a kisebb részébe töltődik be. A látogató itt egy menüvel találkozik és kedvére válogathat a menüpontok között. Több menüpontból, vagyis linkből is választhatunk, amik egy-egy másik oldalra irányítanak át minket. Van itt naptár, hírújság, böngészhetünk a filmek között, illetve itt tekintheti meg a leendő vásárló a bevásárlókosarát. Az alkalmazás rendszergazdája vagy adminisztrátora is ezen a lapon keresztül jelentkezhet be a védett oldalakra. Ezen oldal megvalósításához használunk először űrlapot, ami egy weblap interaktívvá tételének az egyik alapköve.

```
<form method="post" action="belepes.php" target="jobb">
Felhasználónév:<br>
<input type="text" name="felh_nev">
Jelszó:
<input type="password" name="jelszo">
<br>
<input type="submit" name="submit" value="Belépés">
</form>
```

E *form* vagyis űrlap segítségével érhetjük el azt, hogy tudjunk kommunikálni a webszerverrel, tehát dinamikussá tesszük az oldalt. Ezt az űrlapot, mint látható, a belepes.php fájl fogja feldolgozni. Meg kell említenem, hogy a rendszergazda mikor bejelentkezik, a jelszó már eleve titkosítva jut el a távoli webszerverhez.

4. belepes.php

A rendszergazda, illetve adminisztrátor beléptetésére és a rendszergazdai feladatok ellátására szolgáló oldal. Itt tudnak az arra jogosultak új filmeket fölvinni az adatbázisba, illetve filmeket törölni. Ebben a részben használunk először *cookie*-kat, vagyis „sütiket”.

Cookie: egy kisméretű állomány vagy karakterlánc, mely mindig tartalmaz egy nevet, egy értéket, egy érvényességi időt valamint a küldő hoszt nevét és elérési útvonalát. Sütik segítségével, -például korlátozni tudjuk a webhelyünk egyes részeihez való hozzáférést.. Az alábbi kis kódrészletben látható, hogy egy lekérdezés eredményét vizsgáljuk, vagyis, hogy a megadott felhasználói név és jelszó helyes-e. Ha igen, tehát a felhasználónak sikerült belépnie a rendszergazdai oldalra, akkor egy *cookie*-t küldünk a böngészőnek.

```
if (mysql_num_rows($eredmeny) == 1) {
        $vezetek_nev = mysql_result($eredmeny, 0, 'vezetek_nev');
        $kereszt_nev = mysql_result($eredmeny, 0, 'kereszt_nev');
        setcookie("jogos","1",0,"/","",0);
        $uzenet = "<strong>Sikeres belépés...</strong><br>
        <font color= greensize=2> $vezetek_nev $kereszt_nev</font>";
```

A sütink neve „jogos” lesz, értéke pedig 1. Érvényességi időnek nullát adunk meg, ami azt jelenti, hogy a süti csak addig lesz érvényes, ameddig a böngésző adott munkamenete tart. Ha a felhasználó bezárja a böngészőablakot, akkor a süti is érvényét veszti.

5. regisztracio.html

Itt tulajdonképpen egy hírújságra tudnak a látogatók regisztrálni, amelyben a filmvilágból származó legfrissebb híreket tesszük közzé, természetesen e-mailben. Ez csak egy amolyan kiegészítő szolgáltatás a webshop mellé, de ezzel is a vevők megelégedését akarjuk elérni, illetve éreztetni, hogy foglalkozunk velük.

6. naptar.php

Mint a fájl nevéből is kiderül ez egy naptár lesz, melyet azért hoztam létre, hogy a vásárló meg tudja nézni, - mondjuk egy 10 napos szállítási határidővel számolva, hogy mikorra is kapja kézhez a megrendelt dvd filmet. A naptár működése egyszerű, viszont a megvalósításánál számos akadályba ütköztem. Nem volt könnyű létrehozni és működőképessé tenni. Tehát, a felhasználó egy űrlap segítségével választhatja ki az évet és a hónapot, a választott hónap naptára pedig a hét napjai szerinti sorrendben jelenik meg. Két változót használtam, az egyikben az évet, a másikban a hónapot tároltam és ezekből készítettem egy időbélyeget.

Első probléma: Mi történik azokban az esetekben, mikor a felhasználó először lép erre a menüpontra vagy érvénytelen adatokat ad meg vagy nem ad meg semmit? Semmi sem jelenik meg a képernyőn. Ez nem egy szép megoldás így, éppen ezért kell a naptár számára egy alapértelmezett érték. Ez lesz az aktuális év és hónap. Ezt a értéket még az űrlapon a *selected* attribútum segítségével kiválasztom. A szkript elején definiáltam, hogy mit is tekintek „EGY_NAP”-nak. Úgy határoztam, hogy egy napot a másodperceinek számával adom meg. A *checkdate()* függvény arra szolgál, hogy ellenőrizni tudjam, hogy az űrlapon megadták-e az évet és hónapot, illetve érvényes dátumot jelentenek-e ezek. Ha nem, akkor a *getdate()* függvénnyel előállítom az aktuális dátumot és ez lesz az alapértelmezett érték, amivel a naptár megjelenik.

```
<?php
	define("EGY_NAP", (60*60*24));
	if (!checkdate($_POST[honap],1,$_POST[ev])) {
		$mostTomb=getdate();
		$honap=$mostTomb['mon'];
		$ev=$mostTomb['year'];
		}
	else {
		$honap=$_POST[honap];
		$ev=$_POST[ev];
		}
	$kezdet=mktime(12,0,0,$honap,1,$ev);
	$elsoNapTomb=getdate($kezdet);
?>
```

Ha érvényesek az adatok a *mktime()* függvény segítségével egy időbélyeget hozok létre, méghozzá az adott hónap első napjára vonatkozóan. A hónap és a napok neveit egy-egy tömbben fogjuk tárolni, a naptár megjelenítéséhez pedig egy táblázatra lesz szükségünk.

Második probléma: Hogyan döntöm el, hogy a táblázatban egy sornak a végére értem vagy akár az egész naptár végére? Ezt úgy nézem meg, hogy a *$szamlalo* változó 7-tel való osztásának a maradékát vizsgálom. Ha ez nulla vagy a hét többszöröse, akkor a sor vagy a hónap végére értünk.

```
for ($szamlalo=0; $szamlalo < (6*7); $szamlalo++) {
        $napTomb=getdate($kezdet);
        if (($szamlalo%7)==0) {
                if ($napTomb['mon'] != $honap) {
                        break;
                }
        else {
                print "</tr><tr>\n";
                }
        }
```

Harmadik probléma: Hogyan jelenítsem meg a táblázatban a dátumokat? Mert egyáltalán nem biztos, hogy a hónap hétfővel kezdődik, illetve vasárnappal végződik, tehát lesznek üres cellák is a táblázatban. PHP-ban az eredeti számozás szerint a vasárnap jelenti a nullát, míg ugye a magyar naptárakban a hét, a hétfővel kezdődik. Ezért el kell érnünk azt, hogy a hétfő legyen a nulla. Ezt úgy oldottam meg, hogy a *$elsoNapTomb*-ben tárolt hónap első napjának a sorszámához, hozzáadtam hatot, majd vettem ennek az egésznek a héttel való osztásának a maradékát.

```
if ($szamlalo<(($elsoNapTomb['wday']+6)%7)||$napTomb['mon']!=$honap) {
        print "\t<td><br></td>\n";
        }
else {
        print "\t<td>".$napTomb['mday']."    </td>\n";
        $kezdet+=EGY_NAP;
        }
}
print "</tr></table>";
```

A feltételben vizsgálom, hogy *a $szamlalo* értéke kisebb-e az előbb kapott sorszámnál, ha igen, akkor még nem értük el azt a cellát, ahová írni kell, tehát üres cella. A feltétel másik részében pedig azt vizsgálom, hogy még az adott hónapban vagyunk-e, ha nem, akkor a táblázat végére szintén üres cellát hozunk létre. Az *else*

ágban kiírjuk a dátumot, majd megnöveljük a *$kezdet* változó értékét és a ciklus kezdődik előről.

7. kuldes.php

Ennek a fájlnak a segítségével tudjuk elküldeni az előzőekben hírújságra jelentkezett vevőknek az adatait a saját e-mail címükre, illetve rögzítjük az adatbázisban ezeket. Azért küldünk e-mailt a jelentkezőknek, hogy ellenőrizhessék, hogy jól adták-e meg az adataikat. Adatbázisban való rögzítésre azért van szükség, hogy majd a tárolt e-mail címekre el tudjuk küldeni a hírújságot.

Mikor egy HTML-kódba ágyazunk be egy PHP-kódblokkot, akkor jelezni kell a PHP-rendszerrel, hogy hol vannak ezek a parancsok, amelyeket végre akarunk hajtani. Tehát a PHP-kódblokk elejét és végét is jelezni kell és ezt úgynevezett tag-ek segítségével tudjuk megtenni. Ebben a szkriptben már felhasználjuk a PHP nyelv által nyújtott eszközök egy részét is. Nézzük sorjában. A PHP-ban vannak úgynevezett szuperglobális változók, amelyek mindig jelen vannak és mindig elérhetőek. Ilyen szuperglobális változó a $_POST is. Ez azokat a változókat tartalmazza, amelyeket a szkript a POST eljárás révén kapott meg. Tehát a mi $_POST[nev] vagy $_POST[email] változónk, valójában egy hivatkozás egy olyan változóra, amit a $_POST szuperglobális tárol és a „nev”, illetve „email” mezők értékét tartalmazza.

```
<?php
    print "<h2>Kedves <b>$_POST[nev]</b>, köszönjük, hogy regisztrálta
    magát!</h2><br><br>\n\n";
    print "Lakóhelye: <b>$_POST[cim]</b><br>";
    print "Életkora: <b>$_POST[kor]</b><br>";
    print "Az ön e-mail címe: <b>$_POST[email]</b><br><br>\n\n";
        $kapcs = mysql_connect("sql","ocsabus","ff28ee6243");
        mysql_select_db("ocsabus", $kapcs) or die(mysql_error());
        $sql1 = "INSERT INTO hirujsag VALUES (", now(), '$_POST[nev]',
        '$_POST[varos]', '$_POST[eletkor]', '$_POST[email]')";
        $eredmeny1 = mysql_query($sql1, $kapcs) or die(mysql_error());
    $uzenet = "Név: $_POST[nev]\n";
    $uzenet.= "Lakóhely: $_POST[cim]\n";
    $uzenet.= "Életkor: $_POST[kor]\n";
    $uzenet.= "E-mail: $_POST[email]\n";
    $uzenet.= "Jó böngészést kívánok a hírek között!";
    $cimzett = "$_POST[email]";
    $targy = "Regisztráció";
    $fejresz = "X-FW-MailID: fe9c60f5c2";
    mail($cimzett, $targy, $uzenet, $fejresz);
?>
```

Ugyebár a MySQL-t nem kell ugyanazon a gépen futtatnunk, mint a webszerverünket, de a PHP függvényeink csak akkor lesznek képesek a MySQL-lel kommunikálni, ha egy olyan helyen fut, melyhez a webszerverünk csatlakozni tud. Ehhez rendelkeznünk kell az adatbázis használatához egy érvényes felhasználónévvel, egy jelszóval, illetve tudnunk kell annak az adatbázisnak a nevét, amelyikhez csatlakozni kívánunk. A csatlakozás fölépítéséhez egy függvényre lesz szükségünk, mégpedig a *mysql_connect()* függvényre. Ennek segítségével könnyedén, gyorsan és biztonságosan tudjuk létrehozni a kapcsolatot. A *mail()* függvény használatát, majd a megrend_kuldes.php szkript magyarázatánál szeretném kifejteni.

8. bongeszes.php

Ez az oldal szolgál arra, hogy a leendő vásárló „túrkálni” tudjon az adatbázisban levő filmek között. Tehát ezen az oldalon soroljuk föl az összes filmet és paramétereit. Ahhoz, hogy meg tudjuk jeleníteni a filmek adatait, először kapcsolatot kell teremteni az adatbázis-szerverrel. A kapcsolat felépítése után, több *while* ciklus és *select* utasítás segítségével, lekérdeztem az egyes filmek adatait és ezt megjelenítettem a képernyőn. A filmek mellé kiírattam, a rendelni kívánt mennyiséget is, tehát egy filmből nem csak egyet lehet megrendelni egyszerre. Viszont maximum csak annyit, amennyi a raktárkészlet. Szeretném prezentálni egy kis kódrészlettel mindezt, a teljesség igénye nélkül.

```
$sql1 = "SELECT * FROM film";
$eredmeny1 = mysql_query($sql1, $kapcs) or die(mysql_error);

print "<h2 align=center>DVD FILMEK</h2><br><br>";

while ($ujtomb = mysql_fetch_array($eredmeny1)) {
        $film_id = $ujtomb['film_id'];
        $cim = $ujtomb['cim'];
        $megjelen_ev = $ujtomb['megjelen_ev'];
        $jatek_ido = $ujtomb['jatek_ido'];
        $ar = $ujtomb['ar'];
        $keszlet = $ujtomb['keszlet'];
        $kategoria_id = $ujtomb['kategoria_id'];
        $rendezo_id = $ujtomb['rendezo_id'];
        print "Cím:            $cim<br>";
        print "Megjelenés éve:   $megjelen_ev<br>";
        print "Játék idő:       $jatek_ido perc<br>";
        print "Ár:                              $ar Ft<br>";
        print "Raktári készlet:  $keszlet<br>";
        print "Hang: ";
```

9. ujfilm.php

Ezen lap segítségével tud a rendszergazda új filmeket fölvinni az adatbázisba. Az oldal tulajdonképpen egy nagyméretű űrlapból és *insert* utasítások sokaságából áll. Az űrlapban *text* mezőket, *checkbox*-okat illetve *radio* gombokat használtam fel. A *checkbox*-okkal a felirat és a hang nyelveit választhattuk ki, tehát egyszerre több nyelvet is, míg a *radio* gombbal a film kategóriáját adhattuk meg. Az űrlap elküldése után vizsgáltam, hogy ki vannak-e töltve a mezők, ha igen, akkor először felépítettem a kapcsolatot a MySQL-lel és beszúrtam az adatokat a megfelelő táblákba.

Például:

```
$kapcs = mysql_connect("sql","ocsabus","ff28ee6243");
mysql_select_db("ocsabus", $kapcs) or die(mysql_error());

$sql1 = "INSERT INTO film (film_id, film_letre_dat, cim, megjelen_ev, jatek_ido, ar, keszlet, kategoria_id) VALUES ('', now(), '$_POST[cim]', '$_POST[megjelen_ev]', '$_POST[jatek_ido]', '$_POST[ar]', '$_POST[keszlet]', '$_POST[kategoria_id]')";
mysql_query($sql1, $kapcs) or die(mysql_error());
```

10. filmtorles.php

A rendszergazda itt törölheti a filmeket, - például a régieket vagy a már nem forgalmazottakat. Mivel a biztonság nagyon fontos, vagyis ebben az esetben az ,hogy illetéktelenek ne tudjanak erre az oldalra lépni és filmet törölni, rögtön egy *cookie* ellenőrzéssel indítunk.

```
if ($_COOKIE[jogos] == "1") {…}
```

Az általunk a belepes.php oldalon elküldött „jogos” sütink értékét és egyáltalán meglétét ellenőrizzük. Ha jogos az itt tartózkodás akkor a felkínáljuk a filmek listáját, melyekből kiválasztva a megfelelőket, már törölhetünk is. A törlés egyszerűen a kiválasztott filmnek megfelelően, egy *delete* utasítás segítségével történik.

11. kosarba_tesz.php

Ez a szkript arra szolgál, hogy felvegye az adatokat a kosar táblába, majd átirányítsa a felhasználót egy olyan oldalra, amely a kosarának a tartalmát jeleníti meg. A szkript elején rögtön egy munkamenetet indítunk el a *session_start()* függvény alkalmazásával, amit a kosar táblába is be kell írnunk.

Felhasználói munkamenetek (user sessions): A munkamenetek célja az, hogy a felhasználókat egy egyedi azonosítóval lássuk el és így lehetőségük nyíljon, személyre

szabott információkhoz való hozzáférésre is. Amikor a felhasználó letölt egy munkameneteket is kezelő oldalt, akkor vagy egy új azonosítót kap, vagy az előző letöltéskor használt azonosítót kapja meg újra. A munkamenetekhez tartozó változók értéke a letöltések között is megmarad, és a PHP-kódból a $_SESSION szuperglobálison keresztül lesz elérhető. A munkamenet-azonosítókat a kliens gépén alapértelmezés szerint sütikben tárolják, azonban a sütik használata letiltható, ezért ha biztosra akarunk menni, akkor a munkameneteket használó oldalakon az URL-ekbe kell belefűznünk az azonosítókat.

A munkamenet elindítása után ellenőrizzük, hogy a film_id változónak egyáltalán van-e értéke. Ha van, akkor az azt jelenti, hogy a felhasználó a filmek böngészése és a film kiválasztása után jutott el ide. Utána megnézzük, hogy a kapott érték érvényes-e, amit egy SQL lekérdezés segítségével tudunk megtenni.

```
if ($_POST[film_id] != "") {
        $sql1 = "SELECT * FROM film WHERE film_id =
$_POST[film_id]";
        $eredmeny1 = mysql_query($sql1, $kapcs) or die(mysql_error());
        if (mysql_num_rows($eredmeny1) < 1) {
                header("Location: bongeszes.php");
                exit;
                }
else {…}
```

Ha minden rendben van, akkor az *else*-ágban lévő kód segítségével egy *insert* utasítással, elhelyezzük a film adatait a kosar táblában.

12. kosar.php

Az adott felhasználó kosarának tartalmát, ezzel a szkripttel jelenítjük meg. A szkript elején szükséges a már megkezdett munkamenetet folytatni, hiszen az aktuális munkamenet-azonosító alapján fogjuk kikeresni az adatbázisból a felhasználó bevásárlókosarát. Az kosár tartalmának lekérdezése után, mármint ha nem üres, egy táblázatban megjelenítjük a kiválasztott filmek egyes adatait. Itt lehetőséget biztosítunk arra, hogy a már kosárba tett filmet, egy a táblázatban levő link segítségével törölni lehessen.

13. kosar_torles.php

Ebben a szkriptben annyi dolgunk van mindösszesen, hogy végrehajtsunk egy *delete* adatbázis-műveletet, majd átirányítsuk a vevőt egy másik oldalra. Itt biztosítunk lehetőséget a vásárlónak arra, hogy ha meggondolta magát, akkor törölhessen elemeket

a kosarából még a megrendelés elküldése előtt. Itt megint csak folytatjuk a már megkezdett munkamenetet és beazonosítjuk a vásárló bevásárlókosarát.

```
<?php
    session_start();
    $kapcs = mysql_connect("sql","ocsabus","ff28ee6243");
    mysql_select_db("ocsabus", $kapcs) or die(mysql_error());

    if ($_GET[kosar_id] != "") {
            $sql1 = "DELETE FROM kosar WHERE kosar_id =
    $_GET[kosar_id]    AND munkam_id = '$PHPSESSID'";
            mysql_query($sql1, $kapcs) or die(mysql_error());
            header("Location: kosar.php");
            exit;
    }
    else {
            header("Location: bongoszes.php");
            exit;
    }
?>
```

Megvizsgáljuk a $_GET változó értéket, amit a kosar.php oldalról kaptunk és ha ilyen érték nem létezik, akkor a felhasználó nem a kosar.php oldal hivatkozása útján került ide, ezért visszairányítjuk a böngészéshez. Ha van érték és jó, akkor végrehajtjuk a megfelelő SQL-utasítást, vagyis töröljük a filmet, majd a felhasználót visszairányítjuk a kosarához.

14. megrend_urlap.php

Ezen az oldalon folytatjuk a megkezdett munkamenetet, kiíratjuk a vásárló kosarának tartalmát, hogy láthassa, mit és mennyiért rendelt. Egy űrlap segítségével, melyet majd a megrend_kuldes.php szkript fog feldolgozni, bekérjük a megrendelő adatait, nevét, címét, elérhetőségeit.

15. megrend_kuldes.php

Folytatjuk a munkamenetet, kiírjuk a megrendelő és a megrendelés adatait a képernyőre és egy e-mail-t is küldünk a vevőnek. Az adatokat a megrendelesek és a megrend_tetelek táblában tároljuk. A vevő az általunk elküldött e-mail-ben, újra ellenőrizheti a megrendelésének adatait, illetve nyugtaként szolgál a filmek átvételekor. A megrendelő elküldése után, töröljük az adott munkamenethez tartozó

kosár tartalmát. Most szeretném illusztrálni a *mail()* függvény működését, ami talán nem is oly egyszerű és számos buktatót rejthet magában.

A mail() függvény „működés” közben:

```
$uzenet="Név: $_POST[megrend_nev]\n";
$uzenet.="Ország: $_POST[megrend_orszag]\n";
$uzenet.="Irányítószám: $_POST[megrend_irszam]\n";
$uzenet.="Város: $_POST[megrend_varos]\n";
$uzenet.="Cím: $_POST[megrend_cim]\n";
$uzenet.="Telefon: $_POST[megrend_tel]\n";
$uzenet.= "Megrendelt filmek összértéke: $vegossz Ft\n";
$uzenet.="Köszönjük, hogy nálunk rendelt!";
$cimzett="$_POST[megrend_email]";
$targy="Megrendelés";
$fejresz="X-FW-MailID: ff9c60f5c2";
mail($cimzett, $targy, $uzenet, $fejresz);
```

A mail() függvényt csak úgy tudjuk levélküldésre használni, ha előtte beállítottuk a megfelelő direktívákat a php.ini állományban.

Az én beállításaim:

```
[mail function]
; For Win32 only.
SMTP = mail.t-online.hu

; For Win32 only.
;sendmail_from = X-FW-MailID: ff9c60f5c2
```

A *mail()* függvény csak akkor tud levelet küldeni, ha el tud érni egy működő levelezőszervert. Az első direktívában a kimenő levelezésünkhöz használt levelezőszervert írtam be. Míg a második direktívában, a kimenő e-mail feladó mezőjébe kerülő e-mail címet kell beírni. Ez nálam elég furán néz ki, de ezt nem én határoztam meg, hanem a szolgáltató.

4.2. A webáruház dokumentációja

Ebben a részben szeretném ismertetni, illusztrálni az „elkészült” alkalmazás működését és lehetőségeit. Alapigazság, hogy egy alkalmazás sosincs kész. Mindig lehet finomítani, bővíteni, újabb funkciókkal ellátni, ezekre a későbbiekben még kitérek. Először a felhasználóknak, utána pedig a rendszergazdának szánt felületet mutatom be.

Felhasználói felület

Kezdőlap…

Íme a honlapom kezdőoldala. Itt mindenféle bejelentkezés nélkül tudunk, naptárat nézni, feliratkozhatunk hírújságra, böngészhetünk a dvd filmek között, illetve megtekinthetjük a bevásárlókosarunk tartalmát, amennyiben már kiválasztottunk egy-két filmet. Bejelentkezni csak a rendszergazdának vagy az adminisztrátornak kell, mert ők más adatbázis karbantartó funkciókat is elérnek. Az ő felhasználónevüket és jelszavukat, közvetlenül SQL-utasítások segítségével vihetjük be a megfelelő táblába.

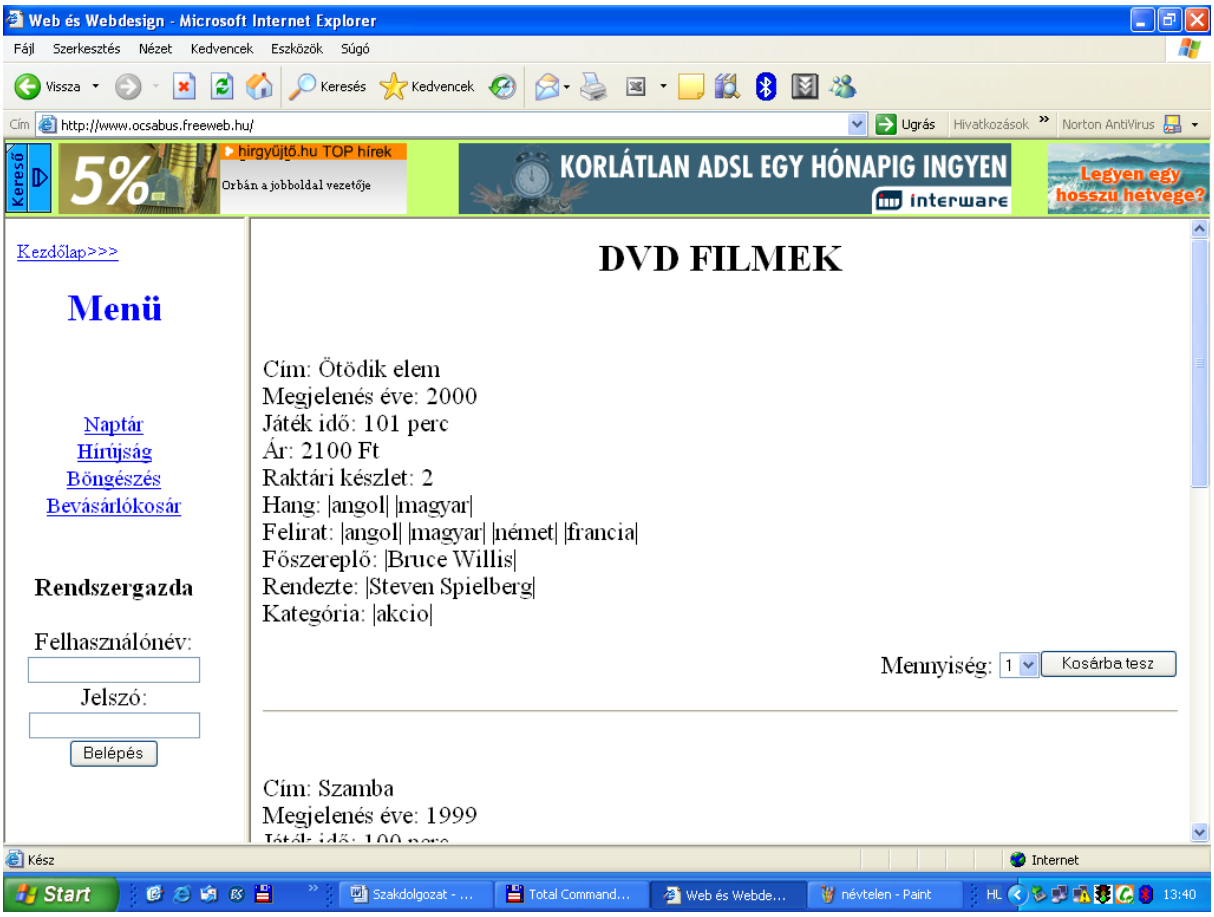
Naptár…

Ez itt a naptár felülete. Meg kell adnunk az évet és hónapot és a „Mehet” gombra klikkelve megkapjuk a tárgyhó naptárját. Ez a funkció hasznos lehet a felhasználónak -például arra, hogy megnézhesse mikorra várhatja a megrendelt dvd filmeket.

A hírújság…

A felhasználó itt tud regisztrálni a hírújságra. Ezen oldal funkciója egyszerű és egyértelmű, a PHP *mail()* függvényének segítségével, a regisztráció után egy e-mail-t kap a felhasználó a regisztrációjának visszaigazolásaképpen, majd minden héten egy e-mail-t kap a legfrissebb filmes pletykákról, hírekről. A hírküldés funkciójának megvalósítását, ez a programverzió még nem tartalmazza.

A dvd filmek listája, vásárlása…

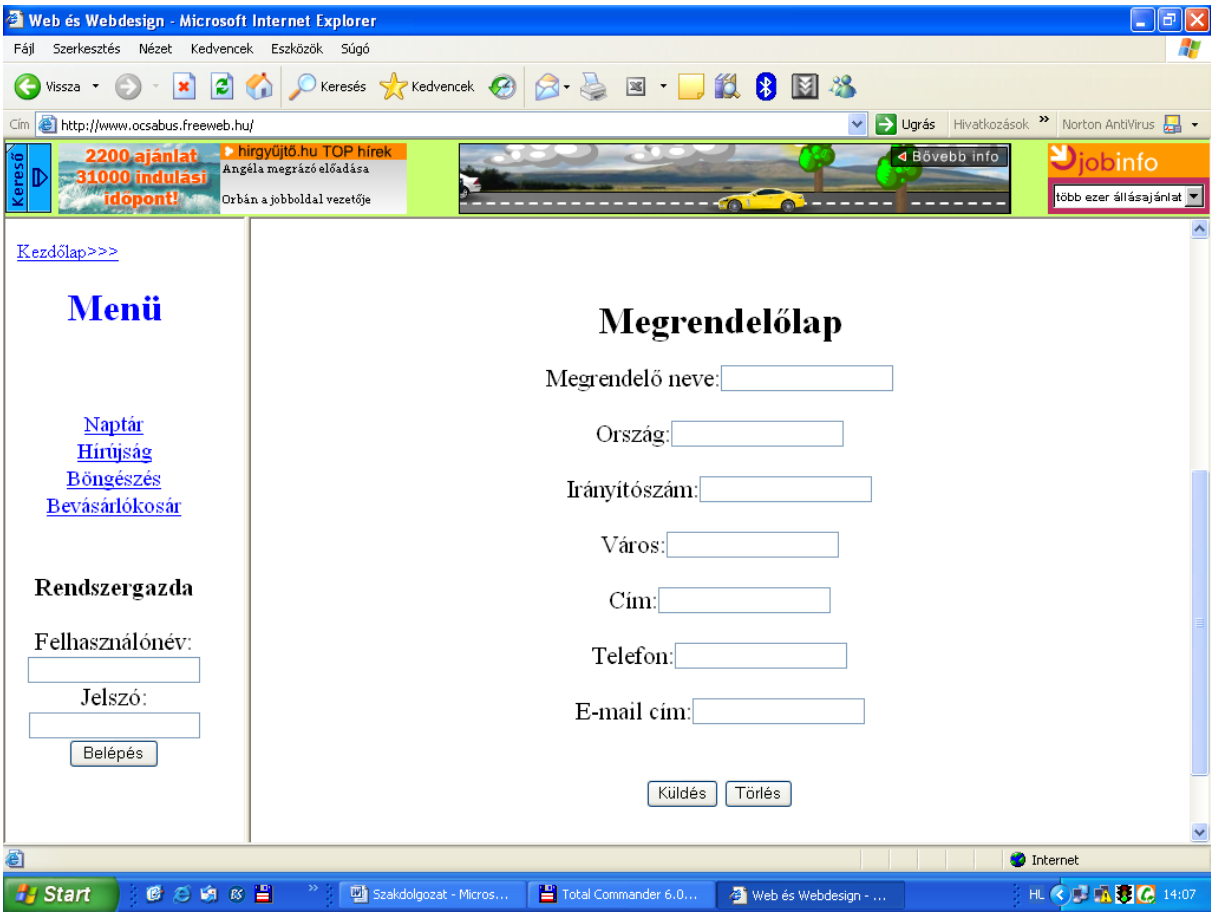


Tehát böngészhetünk a filmek között. Láthatóak a filmek adatai, paraméterei, sőt ki is választhatjuk a megvásárolni kívánt filme(ke)t. Ha úgy dönt a felhasználó, hogy vásárolni kíván, akkor meg kell adnia, hogy hány filmet kíván venni ugyanabból a filmből – alapértelmezésként egy van beállítva - és rá kell klikkelnie a „Kosárba tesz” gombra. Egy vásárlással természetesen több filmet is kiválaszthatunk bármely mennyiségben, pontosabban a raktárkészlet erejéig. Rendelni kívánt mennyiségnek csak a raktáron szereplő filmek darabszámáig kínál fel lehetőséget. Ha rákattintottunk a „Kosárba tesz” gombra, azonnal megmutatja a leendő vevő bevásárlókosarának a tartalmát. Fontos, hogy még semmi elkötelezettsége sincs a vevőnek. De ezt mindjárt látni fogjuk.

A bevásárlókosár…

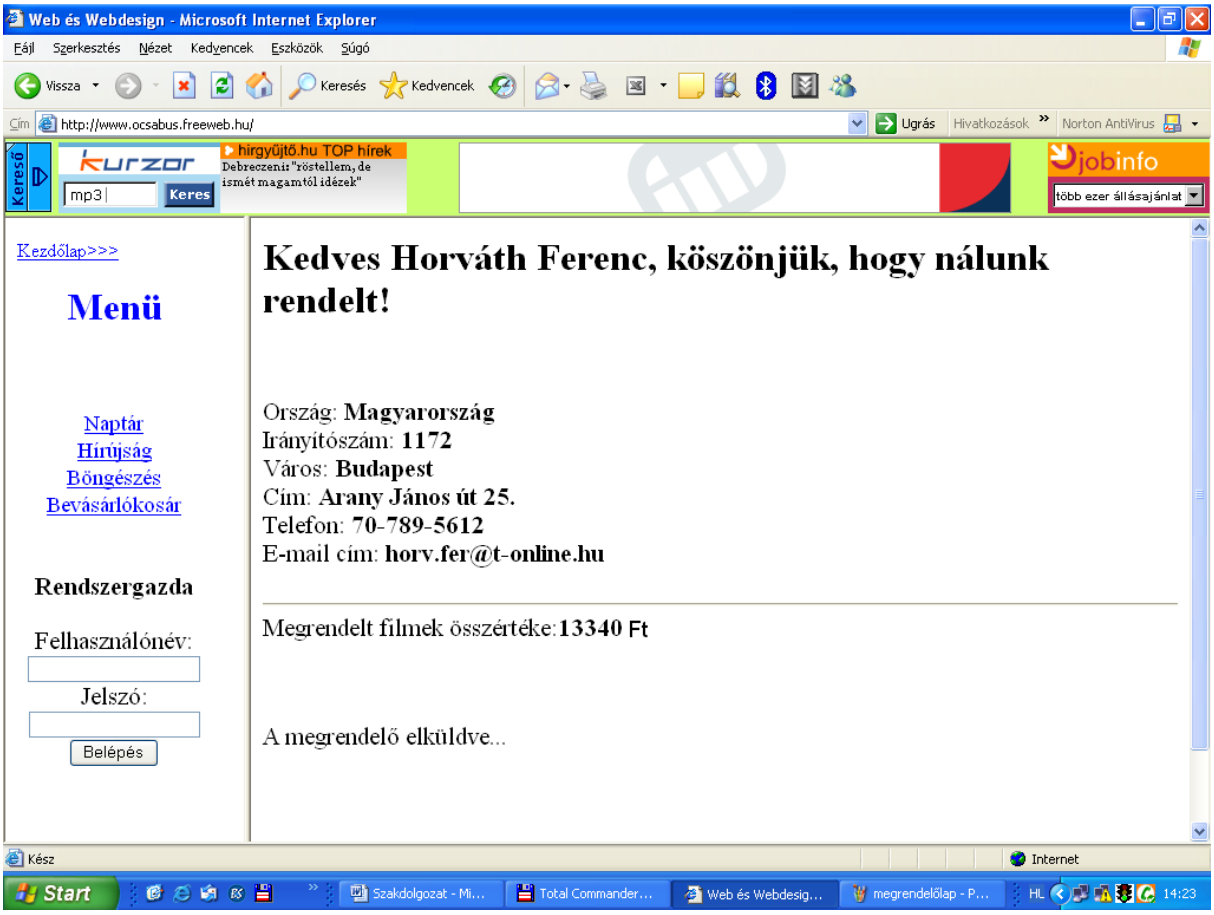
A kiválasztott filmeket tekintheti meg a vevő, egységárral, rendelt mennyiséggel és összesen adatokkal együtt. Látható, hogy itt a vevő még szabadon törölhet a kosarából filmeket, illetve még választhat is másokat hozzá a „Vissza” link segítségével. A „Végösszesen” ténylegesen a megrendelés után kifizetendő díjat jelenti. Tehát – például nincs figyelembe véve a szállítási, postázási költség. Ezzel a későbbiekben még lehet bővíteni az alkalmazásunkat. Ha a vevő úgy dönt, hogy megrendeli a kosarának a tartalmát, akkor a „Megrendelés” linkre kell kattintania. Ha nem rendel, akkor simán törölheti az összes tételét a kosarának és elhagyhatja az oldalt.

Megrendelés…



Az oldal elején - ami sajnos nem látszik, még egyszer megmutatjuk a vevő által megrendelt tételeket, és ha megnézte, kitöltheti a megrendelőlapot majd a „Küldés” gomb megnyomása után a megrendelés rögzítésre kerül az adatbázisban. Ezzel a vásárlás megtörtént. A vásárlás megtörténtéről egy e-mail-t kap a vevő, ami egyben nyugtaként is szolgál a film(ek) átvételekor.

A megrendelés tényének visszaigazolása a képernyőn…



Természetesen az e-mail-en kívül a képernyőn is tájékoztatjuk a vevőt a vásárlás adatairól.

Rendszergazdai felület

A rendszergazdai felületen tudjuk elvégezni az adatbázis karbantartást, a belépéshez felhasználói név és érvényes jelszó kell. Csak ezek megadása után van lehetőség a kívánt műveletek elvégzésére, illetve egyáltalán az oldal elérésére.

Ez a kezdő oldal…

Először is a rendszer azonosítja a rendszergazdát vagy az adminisztrátort, majd két lehetőséget kínál föl:

- film felvitele
- film törlése

A filmek felvitele…

Íme a film felvitelére szolgáló űrlap, ami nem látható a képernyőn teljes egészében. Film felvitele a szövegmezők kitöltésével illetve a *checkbox* - ok és *radio* - gombok kiválasztásával történik.

És végül a film törlésére szolgáló oldal…

A film törlése egyszerű módon, a film címének kiválasztásával és a „Törlés” gomb megnyomásával történik. A film azonnal törlődik az adatbázisból és természetesen, a „Böngészés” menüpont alatt található filmek sorából is.

5. Összegzés

Szakdolgozatom témájaként, egy webáruház megtervezésének, implementálásának a lépéseit szerettem volna bemutatni, és úgy érzem a kitűzött célt sikerült is teljesítenem. Ez a feladat arra is jó volt, hogy kipróbáljam magam, illetve megbizonyosodjak az eddig megszerzett tudásomról. Talán manapság azért is szerencsés volt ezen fejlesztőeszközök segítségével megvalósítani ezt az alkalmazást, mert a webes fejlesztés területén a legelterjedtebbek, könnyen tanulhatóak, illetve a későbbiekben is pénzt hozhat a konyhára a tudásuk. Az alkalmazás létrehozásakor talán annyi előnyöm volt, hogy egy webáruház alapfunkcióit könnyen meg lehet határozni, illetve az egész működéséről hamar meg lehet győződni. A tervezés és implementálás alatt több új ötletem is támadt, amikkel ki lehetne bővíteni a webáruházat. Ezek kivitelezése és megvalósítása után, a rendszerhez integrálásuk könnyedén megoldható.

További ötleteim:

- A felhasználóknak új fizetési módok létrehozása. Eddig a rendszerünkben csak az utánvétes fizetési módra volt lehetőség.
- Rendszergazdák részére különböző lekérdezéseket készíteni, mint – például az ügyfelek listázása, legtöbbet megrendelt filmek listája, már kiszállított, illetve kiszállításra váró filmek stb..
- A már adatbázisban lévő filmek módosításának lehetősége.
- Ügyfelek részére, a legfrissebb filmek elkülönítve legyenek megjelenítve, így könnyebben tájékozódhatnak az újdonságokról.
- Még a hivatalosan meg nem jelent filmekre, előrendelés felvételének a lehetősége.

6. Felhasznált irodalom

Szakkönyvek:

- Julie C. Meloni: A PHP, a MySQL és az Apache használata (Panem, 2003)
- Matt Zandstra: Tanuljuk meg a PHP4 használatát 24 óra alatt (Kiskapu, 2001)
- Julie C. Meloni: Tanuljuk meg a MySQL használatát 24 óra alatt (Kiskapu, 2003)
- Blake Schwendiman: PHP4 Fejlesztők kézikönyve 1-2. kötet (Panem, 2001)
- Peter Moulding: PHP Haladóknak – Fekete könyv (Pervact-Pro, 2002)

Internetes források:

- Apache Software Foundation – Apache Manual: www.apache.org
- PHP Documentation Group – PHP manual: www.hu.php.net
- MySQL AB – MySQL Reference Manual: www.mysql.com
- Programozási fórum: www.prog.hu