Debreceni Egyetem

Informatika Kar

Webes kerékpárkölcsönző implementálása

Témavezető:

Mecsei Zoltán Pál

Egyetemi Tanársegéd

Készítette:

Komlósi Dávid

Gazdaságinformatikus

Debrecen, 2010

Tartalomjegyzék

1. Bevezetés

Szakdolgozatom témájaként olyan területet szerettem volna választani, melyben létrehozhatok egy olyan egyedi produktumot, amelyet azután fel lehet használni gyakorlati célokra. Ezért esett választásom saját program készítésére. Az általam implementált program egy jelenleg fiktív a Debreceni Egyetem részeként működő kerékpárkölcsönző részére készült. A kerékpárkölcsönző, egy az Egyetem által nyújtott szolgáltatás lenne, melyet nem kizárólagosan csak az Egyetem tagjai vehetnének igénybe, hanem a Debreceni lakosság és az idelátogató turisták is. A kölcsönző betöltené azt a piaci rést, melyet az egyre növekvő mozgáskultúra, hiányos tömegközlekedés és egyre növekvő üzemanyagárak mellett nagyban támogatnak továbbá az egyre növekvő kerékpárút beruházások és ezen tényezők mellett Debrecen városa nem rendelkezik megfelelő kerékpár kölcsönzési lehetőséggel.

A kerékpárkölcsönző fő profilja a különböző egyetemi campusok területén elhelyezett telephelyek lesznek, melyek egy tulajdonoshoz tartoznak és lehetőséget nyújtanak a városon belüli kerékpáros közlekedésre azon okból kifolyólag, hogy bármely telephely használható kölcsönzésre és leadásra is egyaránt, nem kötelező a felvétel helyén leadni a kerékpárt. Ezzel az üzleti profillal egy olyan informatikai rendszerre van szükség, mely egyszerre tudja kezelni a több telephely adatait, de mégis külön felülettel rendelkezik mindegyik számára.

A kerékpárkölcsönző működéséhez szükséges egy olyan nyilvántartórendszer, mely a bemutatott több telephely miatti problémát képes megoldani és a kölcsönző működése közbeni adatok tárolása és kezelése megoldható vele. A piacon lévő nyilvántartórendszer szoftverek többsége nem ingyenes licenc alatt használható és nem specifikálható az adott cég igényei szerint. Az általam készített rendszer ingyenes szoftverek felhasználásával készült. Egy adott kölcsönzői profil szerint implementáltam, de könnyen átalakítható és bővíthető kódot hoztam létre, mely segítségével tetszőleges számú és elhelyezkedésű telephely kezelésére alkalmas rendszer áll az Egyetem rendelkezésére.

A nyilvántartórendszer létrehozására egy adatbázisháttérrel rendelkező dinamikus weboldalt választottam, mely a közös adatbázisnak és az egy időben több felhasználó által való elérhetőségének köszönhetően alkalmas a több telephely kezelésére. Ezen weboldal implementálására a PHP és MySQL eszközeit használtam, melyekkel a programozás labor 1 tárgy keretein belül a témavezetőm ismertetett meg. A MySQL adta az adatbázisrendszert a

programhoz, amíg a PHP szkriptnyelv segítségével tudtam azt kezelni és összekapcsolni a weboldallal.

A szakdolgozat célja bemutatni a kerékpárkölcsönzőt, a felhasznált programokat egy olyan szinten, hogy a programból származó kódok értelmezhetőek legyenek előzetes ismeret nélkül. Továbbá az elkészült nyilvántartórendszer bemutatása, felépítés és funkciók szerint, melyek ismeretében az olvasó képes lesz kezelni a programot és a megfelelő programozási ismeret birtokában könnyen testre szabni azt.

2. Felhasznált szoftverek bemutatása

2.1 Mi az a PHP?

PHP jelentése: Hypertext Preprocessor. A PHP egy széles körben használt nyílt forráskódú szkriptnyelv, mellyel dinamikus weboldalakat hozhatunk létre. Az egyik legnagyobb előnye, hogy több operációs rendszeren és többféle web szerveren is futtatható. Az Apache web szerver legnépszerűbb beépülő modulja a PHP.

A leginkább szerver oldalon használt PHP-nak létezik, parancssori interfésze és önálló grafikus felülettel rendelkező alkalmazások is létrehozhatók vele. A PHP egy szerveroldali parancsnyelv, amit leggyakrabban HTML oldalaknál használnak. A PHP parancsok a HTML dokumentumban szerepelnek, de ezeket jól ellehet különíteni egymástól, mellyel a weboldal funkcióinak és megjelenésének fejlesztése jól szétválasztható egymástól. Ezzel kiváló lehetőséget biztosít a weboldal fejlesztésére, mivel a már meg lévő PHP parancsokat érintetlenül hagyva egy teljesen új grafikus felületet hozhatunk létre, vagy a grafikus felületet hagyhatjuk érintetlenül, amíg a teljes PHP utasításkészletet lecserélhetjük, egy jobban optimalizált változatra.

A HTML utasításokkal ellenben a web szerver nem küldi el a PHP parancsokat a felhasználó gépének, hanem azokat a saját PHP motorja futtatja le és csak a parancsok kimenete jelenik meg a végfelhasználó gépén. Leggyakrabban adatbázisok megnyitására, kezelésére, fájlokkal való munkára, tehát a weboldal dinamikussá, interaktívvá tételéhez használják fel. De ezeken felül a PHP ellátja azokat a feladatokat, amelyeket a HTML nem vagy csak részben képes ellátni. Ilyenek a kódolás, adategyeztetés, e-mailküldés, dinamikus listakészítés stb. Ezen felül

minden olyan feladatnál ahol nagyszámú ismétlődő feladatot kell végrehajtani, például listakészítés, a PHP nagyban megkönnyíti a munkánkat. Az 1990-es években az internet hajnalán a weboldalak passzívak voltak. Ekkor még nem volt szükség a PHP- hoz hasonló szkript nyelvekre, de a web fejlődésével és az adatok rohamos növekedésével egyre inkább kialakult az igény erre. A Web 2.0 megjelenésével a tartalmak szolgáltatását a felhasználók is a kezükbe vették, és a weboldalakkal szemben támasztott legfontosabb követelmény a dinamikusság lett. Ma már elképzelhetetlen egy olyan hírportál ahol az olvasók ne kommentezhetnének, azaz kifejezhetnék a véleményüket. Ezen változások létrejöttéhez többek közt a PHP is hozzájárult. A PHP-t lehetséges parancssori alkalmazásként is telepíteni, így alkalmazható kiszolgáló oldali parancsfájlok készítésére is, ami lehetőséget ad például a rendszergazdáknak egyes szerverfeladatok automatizálására.

2.2 A PHP fejlődése

Rasmus Lerdorf 1995-ben alkotta meg a nyelvet, amelyet azóta is folyamatosan fejlesztenek. Akkoriban a PHP: Personal Page Tools néven vált ismertté. Napjainkban a PHP implementációt a PHP Group fejleszti és adja ki a PHP License alatt, ami egy ingyenes licence. Lerdorf a kezdetekben Perl szkriptek lecserélésére írta a PHP-t, ami akkoriban még csak CGI programok halmaza volt. Ezeket kombinálta a szintén általa írt űrlapértelmezővel (Form Interpreter) így hozva létre a PHP/FI-t. 1995. június 8-án hozta nyilvánosságra a PHP első verzióját, abból a célból, hogy a közösség segítséggel gyorsabban lehessen a hibákat javítani és fejleszteni a programot.

A PHP 2-es verziójában már megtalálhatóak voltak a mai PHP-ra is jellemző tulajdonságok, mint az űrlapok kezelése, a HTML kódok beszúrásának lehetősége, a Perl-éhez hasonló változók. Az adatbázisok kezelése már az első verziónak része volt, és ennek segítségével már a kezdetekben is képes volt egyszerűbb dinamikus oldalak létrehozására, Perl használata nélkül.

1997-ben két Izraeli programozó, Zeev Suraski és Andi Gutmans, újraírták az addigi értelmezőt, ezzel létrehozva a kettes és hármas verzió alapját. Ekkor kapta meg a Hypertext Preprocessor nevet a PHP. A hármas verzió sikeres tesztje és kiadása után ismét elkezdték újragondolni a PHP motorját. 1999-ben született meg a Zend Engine és a fejlesztők megalapították a Zend Technologies-t, ami a mai napig részt vesz a PHP fejlesztésében.

2000 májusában jelent meg az immár Zend Engine alapú PHP 4 első verziója. Melynek a fejlesztését, és további támogatását, biztonsági frissítések készítését 2008-ban fejezték be. 2004-ben adták ki a PHP5–öt, melynek az új Zend Engine II. volt az alapja. A legnagyobb fejlesztés, az objektum orientált programozhatóság létrehozása volt. Ezek mellett sok más újítás is szerepet kapott az új verzióban. Többek közt megtalálható az XML támogatottság, és különféle XML kezelő függvények és osztályok kaptak helyet ebben a verzióban. Az új verzió önmagában tartalmazza az SQLite-ot a hozzá tartozó összes függvénnyel együtt, így egyszerűbb adatbázis feladatokhoz nincs szükség külön adatbázis telepítésre, elég egy egyszerű PHP5. Az új Zend Engine az objektumorientált programozhatóságon túl jelentősen növelte a PHP teljesítményét, így összességében alkalmassá téve azt napjaink problémáinak megoldására és a felmerülő igények kielégítésére.

A jelenlegi legfrissebb verzió a PHP 5.3.3, amely 2010 júliusában jelent meg. A fejlesztés folyamatosan zajlik, jelenleg a teljes Unicode támogatottságot próbálják elérni, amelyet még nem tudni, hogy egy teljesen új hatos, vagy egy nagyobb ötös verzióban fognak kiadni.

2.3 A PHP használata

A PHP-t általában HTML dokumentumokba beágyazva használjuk. Sajátossága, hogy nem a felhasználó gépén fog lefutni, mint a HTML utasítások, hanem a web szerveren. A PHP kódrészletek elkülönítése a HTML dokumentumokban többféleképpen is történhet. Az ajánlott jelölés a „<?php” nyitó szkript és „?>” záró szkript használata, mivel ezen jelöléseket a PHP további konfiguráció nélkül is képes értelmezni. További jelölések:

- Rövid címke: <? … ?>
- ASP címke: <% … >
- Szkript címke: <SCRIPT LANGUAGE=”PHP”> … </SCRIPT>

A PHP fájl futatásakor a futató a kimentre másol minden karaktert, amíg nem találkozik a fent említett nyitó szkriptek egyikével. Ekkor lefuttatja a nyitó és záró szkriptek között található kódot és annak eredményét írja a kimentre. Tehát a HTML dokumentum többi része teljesen érintetlenül marad, amíg a PHP kódok lefutásra kerülnek. Így teljesen beágyazható a PHP kód a HTML dokumentumba, annak bármilyen módosítása nélkül.

A PHP, mint a legtöbb programozási nyelv lehetőséget biztosít változók, konstansok, operátorok, függvények, vezérlési szerkezetek és az 5-ös verzió óta osztályok és objektumok használatára is. A PHP szintaktikáját rengeteg könyv, cikk, tutorial oldal és videó segít elsajátítanunk, akár magyarul akár az általunk választott idegen nyelven. A következő példakód segítségével bemutatom egy egyszerű PHP kód működését.

```
<HTML>
   <BODY>
   <b>Matematikai művelet:</b>
   <?            //a „//” után található rész a komment tehát nem kerül értelmezésre
         $szam1=5;     //egy egyszerű változó deklaráció és értékadás
         $szam2=31;
         $eredmeny = $szam1+$szam2; //művelet változók segítségével
         if ($eredmeny=36) print "5 + 31 = $eredmeny";
         //if függvény használata és az eredmény kiíratása
   ?>  //a PHP kód lezárása innentől ismét HTML parancsok szerepelnek
   <br><br>
   <b>A pontos idő:</b> <?= date("H:i:s")
   //újabb PHP rész, itt a date függvénnyel lekérdezzük a pontos időt, ez a szerveren található
   időt fogja visszaadni az általunk megadott formátumban
   ?>
   </BODY>
</HTML>
```

Nincs lehetőségem a szakdolgozat keretein belül bemutatni a PHP nyelv teljes funkciókészletét, de a példakódon is látszik, hogy a PHP mennyire kibővíti a weboldalunk készítésénél az eszközeink tárházát. A PHP nyelv további bemutatása az SQL fejezetben történik, ahol az adatbázis kezelést mutatom be PHP segítségével.

2.4 Az adatbázisokról

A XX. század elején az egyre növekvő adatmennyiség által gyorsan fokozódó igény alakult ki, az adatok gyors gépesített tárolására. Az akkoriban még lyukkártyás gépek, nem voltak alkalmasak jelentős mennyiségű adatok gyors tárolására és visszakeresésére, de az akkori rendszereket is adatbázisnak nevezzük. Az 1960-as években kezdtek kialakulni az úgy nevezett hálós adatbázis rendszerek melyeknek a legnagyobb képviselője a CODASYL volt. A hálós adatmodellben még nem válik szét a fizikai és logikai adatbázis. Az első olyan adatmodell melyben ez a kettő szétválik, a relációs adatmodell. Edgar F. Codd 1969-ben alkotta meg a relációs adatbázis logikai modelljét, de csak 1970-ben publikálta. A relációs

modell elterjedését nagyban gátolta az a tény, hogy a lekérdezések megírásához logikai és halmazelméleti ismeretre volt szükség, mellyel a felhasználók többsége nem rendelkezett. A CIA Larry Elisson és Bob Miner alkalmazásával létrehozott egy projektet. Ennek a projektnek a célja, egy olyan adatbázis létrehozása, mely gyorsan és hatékonyan tudja kiszolgálni a CIA-t. A projekt neve ORACLE volt. A projekt pénzhiány miatt félbeszakadt, de Larry Elisson és Bob Miner kiválva megalapították a saját cégüket melynek neve Relational Software Inc. később Oracle Corporation, melynek keretein belül létrehozták az első Oracle-t.

A lekérdezőnyelv problémáját az Oracle sem oldotta meg. Ezt az első SQL-nek sikerült, melynek alapjait az IBM fektette le, de ő maga nem volt érdekelt akkoriban az adatbázisokban így az első SQL implementáció nem az IBM nevéhez fűződik. Az első forgalomban kapható SQL alapú adatbázis kezelő szoftver az Oracle V2 volt, melynek megjelenése pár héttel megelőzte az eredeti ötletet megalkotó IBM saját rendszerének a megjelenését. 1986-ban Európában és az Egyesült Államokban is szabvánnyá vált az SQL, mint relációs adatbázisokhoz készült lekérdező nyelv. Napjainkban a legelterjedtebb adatbázis modell a relációs modell, mely teljesen kiszorította a különböző hálós- és hierarchikus adatmodellen alapuló adatbázisokat. Az objektumorientált programozási nyelvekkel egyidejűleg megjelentek az objektumorientált adatbázisok, de amennyire az előbbi elterjedt, annál kevésbé nyertek teret az objektumorientált adatbázisok. Az objektumorientált programozási nyelvekkel lehetőség van kezelni relációs adatbázisokat, a logikai adatbázis elve miatt. A legújabb SQL szabvány melyet neveznek OQL-nek és SQL99-nek is, már tartalmazza ezt az elérési módot.

Az adatbázisok használatának elterjedésével és a weboldalak fejlődésével lehetővé vált a webes adatbázisok használata. A web korai szakaszában mikor az általános a passzív, azaz információközlő weboldalak voltak, még nem kellett adatokat beolvasni, kezelni, tárolni. Elég volt a weboldal forrását elhelyezni a szerveren és a felhasználók csak megtekintették az oldalt, nem voltak arra hatással. Napjainkban az oldalak aktívak, azaz a felhasználó nem csak olvassa, hanem bizonyos mértékben befolyásolja is az oldalt. Erre nagyszerű példa a web áruházak, ahol lehet egyszerűen böngészni, de az oldal igazi felhasználásához már be kell jelentkeznünk, ami adatbázisok és szkript nyelvek segítségével történik. Maga a HTML oldal mindössze egy grafikus keretet biztosít az adatbázisból kapott adatoknak. Erre példa az említett web áruházak árukészlete, melyet a weboldal az adatbázisból kérdez le és azt helyezi el a grafikus keretbe. Tehát láthatjuk, hogy egy mai weboldal létrehozásához már közel sem elég az egyszerű HTML. Lehetséges ilyen oldalakat létrehozni, de azok nem fognak

rendelkezni a megfelelő funkciókkal vagy jelentősen lassabbak, nehezebben fejleszthetőek lesznek.

Láthatjuk, hogy az adatbázisok használata mennyire elterjedt napjainkban. Lehetőséget nyújtanak a megnövekedett adatmennyiség gyors, hatékony rendezésére, tárolására és azok visszakeresésére. Ezért is esett a választásom egy adatbázis háttérrel rendelkező dinamikus weboldal fejlesztésére, melyet nyilvántartó rendszerként lehet majd használni. Az általam megvalósított implementációhoz a már fent bemutatott PHP nyelvet használtam, mely a weboldal funkcióinak működését és az adatbázissal való kapcsolatért felel. Adatbázis kezelőnek a MySQL rendszert választottam, többek közt azért, mivel a PHP-hoz hasonlóan freeware, azaz ingyenes szoftverről van szó.

2.5 MySQL

A MySQL egy SQL alapú relációs adatbázis kezelő rendszer (RDBMS-relational database management system). A MySQL eredeti fejlesztője a MySQL AB cég volt, melyet először a Sun vásárolt fel, aztán pedig a Sun-t vásárolta fel az Oracle Corp., így napjainkban a MySQL tulajdonosa az Oracle. Kétféle licenc szerint lehet felhasználni a MySQL-t. Többnyire ingyenes, azaz szabad felhasználású szoftverként funkcionál (GPL-General Public License), de létezik pár specifikus eset, melyben fizetni kell a használatért. Az egyik ilyen, melyben saját programunk részeként használjuk fel a MySQL-t így értékesítve a saját szoftverünket.

A MySQL az egyik legelterjedtebb adatbázis kezelő rendszer, melynek oka többek közt a már említett GPL licenc, de meg kell említeni, hogy a MySQL támogatja a különböző operációs rendszereket, a legtöbb web szervert és emellett a legtöbb programozási nyelv segítségével lehet kezelni. A PHP-MySQL-Apache hármas nagyszerű freeware szoftverhármas, melynek segítségével létrehozhatjuk a saját dinamikus adatbázis alapú weboldalunkat. A MySQL főbb képességei a teljesség igénye nélkül:

- ANSI SQL99 használata, kiegészítésekkel
- Cross-platform támogatás
- Tárolt eljárások
- SSL támogatás
- SELECT-ek egymásba ágyazásának lehetősége
- Részleges UNICODE támogatás
- View készítés

- Beágyazott adatbázis könyvtár
- Lekérdezési gyorstár
- Többféle, többek közt natív tároló motor
- Közösség vagy cégek által fejlesztett tároló motorok
- Lehetőség saját tároló motor használatára

A MySQL SQL lekérdezőnyelvet használ, így néhány különbségtől eltekintve, a MySQL utasításkészlet megegyezik az SQL utasításkészlettel. A többi lekérdező nyelvvel megegyezően, az SQL utasításkészletét is két részre bonthatjuk. Az adatdefiníciós (Data Definition Language - DDL) és adatkezelési (Data Manipulation Language - DML) részekre. Az adatdefiníciós utasítások:

- CREATE utasítás, mely segítségével adatbázisok és táblákat hozhatunk létre.
- ALTER utasítás, mellyel az adatbázis objektumait módosíthatjuk
- DROP utasítás, mely adatbázis objektumokat töröl

Az adatkezelő utasítások:

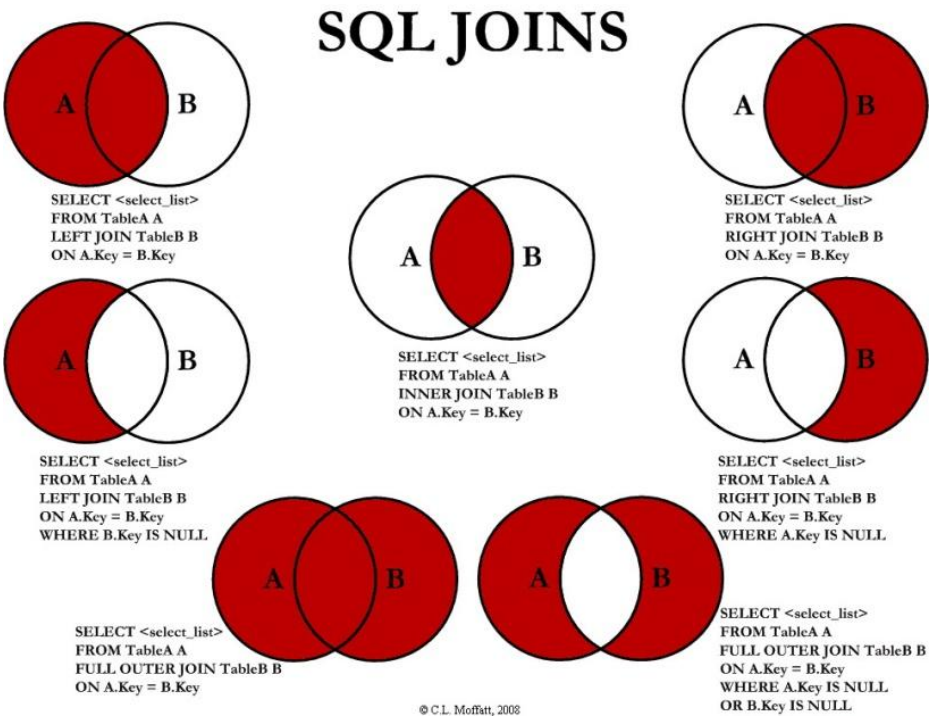
- SELECT a leggyakrabban használt utasítás, a lekérdezés, melynek nagy előnye, hogy a lekérdezésen belül további allekérdezéseket helyezhetünk el több SELECT használatával
- FROM segítségével választhatjuk ki, mely táblában akarunk keresni
- WHERE utasítás segítségével szűrési feltételeket fogalmazhatunk meg a lekérdezésünkben, így leszűkítve a lekérdezés eredményét a számunkra fontos adatokra
- GROUP BY a lekérdezés eredményében lévő adatokat tudjuk csoportosítani az általunk megadott feltétel szerint
- HAVING a már csoportosított eredményekre alkalmazhatunk szűrési feltételek
- ORDER BY az eredményhalmazt rendezhetjük az általunk választott módon

További utasítások:

- INSERT adatok hozzáadását végezhetjük el vele táblák esetén, így hozva létre új rekordokat
- UPDATE a tábla már meglévő adatain végezhetünk módosításokat, az ALTER utasításhoz hasonlóan, melyet az adatbázis elemein használhatunk
- DELETE utasítással adatokat törölhetünk a táblából

Az adatbázisokból történő lekérdezések esetén gyakran előfordul, hogy az általunk felhasználni kívánt adatok több táblában helyezkednek el. Ekkor az SQL lehetőséget biztosít a megfelelő kulcsok kapcsolatán keresztül a táblák összekapcsolására. Ez a művelet nem

befolyásolja a letárolt adatainkat, mindössze logikailag hozza létre a táblák kapcsolásával az általunk kiválasztott új táblát, melyben már lefuttathatjuk a kívánt lekérdezést. Több összekapcsolási lehetőség is rendelkezésünkre áll. A lefuttatni kívánt lekérdezésnek megfelelően kell kiválasztanunk azt a logikai összekapcsolást, mellyel az általunk várt eredményt kapjuk.



2.6 MySQL kezelés PHP segítségével

Az adatbázis kezelése előtt létre kell hoznunk az adatbázis, mely többféleképp is történhet. Létrehozhatjuk az adatbázist PHP fájl segítségével, melyet elég egyszer lefutatni és létrejön az adatbázis. De használhatjuk a különféle MySQL-hez készült adminisztrációs eszközöket melyekkel könnyedén létrehozhatjuk az általunk kívánt adatbázis struktúrát. A legelterjedtebb ilyen adminisztrációs eszközök: MySQL Administrator, MySQL Query Browser és a legnépszerűbb a nyílt forráskódú PHP nyelven írt phpMyAdmin. Ezek az eszközök könnyen kezelhető grafikus felületet nyújtanak a felhasználóknak, melyek segítségével könnyen létrehozhatók és kezelhetők a MySQL adatbázisok.

A már létrehozott adatbázisok kezelése PHP segítségével mindig a kapcsolódással kezdődik. A munkafolyamat előtt az első lépés a kapcsolat létrehozása az adatbázis és a PHP forrás

között. A PHP rendelkezik egy tulajdonsággal, mely lehetővé teszi a feladatok külön állományba történő megoldását. Bármely PHP állományban meghívhatunk egy másik PHP állományt, ez az INCLUDE paranccsal történik. Ekkor a meghívott PHP fájlban található utasítások lefutnak az általunk eredetileg elindított fájl utasításai mellett. Így az összes PHP oldalunkhoz szükséges kapcsolódást megoldhatjuk egyetlen fájlban és a többi helyen ahol szükségünk lesz, rá egyszerűen meghívjuk majd az include parancs segítségével. Ez a módszer lehetővé teszi a weboldalunk könnyebb hordozhatóságát, mivel ha egy új adatbázist akarunk használni alatta, vagy egy új szerverre helyezzük, elég a kapcsolat fájlt átírni és nem kell az összes PHP dokumentumban kikeresni a kapcsolódás részt. A kapcsolódáshoz létrehozzuk egy php állományt és elhelyezzük benne a következő kódot:

```
<?$kapcsolat = mysql_connect("localhost","userid","jelszo");
mysql_select_db("teszt", $kapcsolat);?>
```

A kapcsolat változóban eltároljuk a mysql_connect függvény eredményét, melynek 3 paramétere sorban a következők: a szerver neve, a felhasználónév és jelszó mellyel csatlakozunk. A kód második sorában a mysql_select_db paranccsal kiválasztjuk az általunk használni kívánt adatbázis, jelenleg ez a teszt nevezetű adatbázis, és megadjuk a használni kívánt kapcsolatot, ez a már letárolt kapcsolatváltozó segítségével történik. Innentől, ha ezt az állományt meghívjuk bármely PHP kódunkban, akkor onnantól dolgozhatunk az adatbázissal, mivel már megteremtettük a kapcsolatot. A kapcsolatot a mysql_close($kapcsolat) paranccsal lehet lezárnunk a kód legvégén, igaz hogy a kapcsolat automatikusan lezárul, ha végzett az összes utasítással, de ha kinyertük a szükséges adatokat ajánlott lezárni a kapcsolatot manuálisan.

Az adatbázissal való munkához a már megismert SQL utasításokat tudjuk használni. Ezeket szintén PHP változókban tároljuk és a megfelelő függvények meghívásával tudjuk lefutatni az adatbázison. A PHP kódban a mysql_query() függvény szolgál az SQL lekérdezések futtatására. A függvény egy kötelező és egy opcionális paraméterrel rendelkezik. A kötelezően megadandó paraméter a futatni kívánt SQL utasítás. Az opcionális paraméterként pedig megadhatjuk a használni kívánt kapcsolatot, ha nem adunk meg semmit, akkor az utoljára használt kapcsolatot fogja használni. Az opcionális paramétert elhagyhatjuk, ha egy adatbázissal dolgozunk, tehát az aktív kapcsolatunk mindig a kapcsolódásnál már deklarálódik. Ezzel a módszerrel olyan utasításokat lehet kezelni, melyek az adatbázison hajtódnak végre és nem adnak számunkra fontos adatokat vissza, azaz a create, insert, delete stb. parancsok.

```
<?$parancs = „INSERT INTO teszt(teszt, elek)”;
mysql_query($parancs);?>
```

Látható, hogy nem kezelünk visszatérési értéket, egyszerűen lefutatjuk a parancsot az adatbázison, mely jelen esetben a teszt táblához ad egy új sort, melynek értékei teszt és elek.

Ez a módszer az adatkezelő utasításoknál a jelen formájában nem alkalmazható, mivel ott a lekérdezett adatok fontosak számunkra, azaz tárolni kell őket a lekérdezés lefutása után. A lekérdezéshez használt függvény szintén a mysql_query(), melynek paramétereként most egy SELECT utasítást adunk meg. Ekkor a mysql_query függvény egy eredménytáblát fog visszaadni, melyben az általunk kért adatok szerepelnek, ez tábla logikai tábla lesz, tehát nem lesz az adatbázis részeként letárolva. Az előző példakódhoz képest mindössze annyit módosítunk, hogy a mysql_query() függvényt eredményét futatáskor egy változóba eltároljuk. $eredmeny=mysql_query($parancs);

Ekkor az $eredmeny változó segítségével kinyerhetjük az általunk lekérdezett adatokat. Erre többféle függvényt is használhatunk. A mysql_num_rows() függvény használatával meghatározzuk az eredménytábla sorainak számát. Az eredménytábla feldolgozásához több függvény is rendelkezésünkre áll. Az egyik a mysql_fetch_row(), mely az $eredmeny változót megkapva, sorról sorra kiolvassa az adatokat a táblából, és ha elfogytak, akkor false értékkel tér vissza. A mysql_fetch_array() egy tömböt ad vissza, melynek értékei az eredménytáblában szereplő adatok. Ezt a függvényt használják leggyakrabban, mivel könnyű letárolni vele a kapott adatokat és a sorból kapott információkat szűrhetjük is. Létezik még a mysql_fetch_object(), amellyel egy objektum tulajdonságaiként férhetünk hozzá a mezőkhöz. Ekkor a tulajdonságok nevei lesznek a mezőnevek. Ez a módszer a PHP5 verziójától érhető el, melyben implementálták az objektumorientált programozás lehetőségét.

Az általam bemutatott módszerek közel sem fedik le a teljes rendelkezésünkre álló funkciókészletet, de a készített program értelmezéséhez szükséges alapinformációk szerepelnek benne. A program bemutatásában a programkódok részletezésekor a még nem ismertetett függvények, parancsok bemutatásra kerülnek.

3. Kerékpárkölcsönző háttere

3.1 A „cég” leírása

A bevezetőben már említett szolgáltatás, egy kerékpárkölcsönző, amely a Debreceni Egyetem részeként vagy egy külső cégként is képes lenne működni. A Debreceni Egyetem részeként működhet a kölcsönző, mint szolgáltatás a hallgatók részére és elérhető szolgáltatás azon emberek részére, akik nem tagjai az egyetemnek. Külső cégként történő funkcionálása megoldható, ha a Debreceni Egyetem szolgáltatja a szükséges erőforrásokat (bővebben a Rendelkezésre álló erőforrások c. fejezetben). Ezen esetben egy olyan szolgáltatás jönne létre az egyetem területén, melyből mind az egyetem hallgatói, mind Debrecen lakosai profitálhatnának. Erre jó példa az egyetem területén elhelyezkedő bankok és Posta. Jelen szakdolgozat azon lehetőséget vizsgálja melyben a kerékpárkölcsönző a Debreceni Egyetem részeként működik és igénybe vehető szolgáltatást nyújt mind az egyetem tagjai és mind a további lakosság részére.

Debrecen városában a folyamatos fejlesztések nyomán egyre több kerékpárút épül. A Debreceni Egyetem is kilátásba helyezte egy az egyetemi campusokat egybekötő kerékpárút megépítését. Ezzel egyidejűleg nem létezik egy olyan kerékpárkölcsönző, amelyet igénybe lehetne venni, olyan hallgatóknak, akik vagy nem rendelkeznek kerékpárral vagy nem Debreceni lakosok, így nem szállítják a kerékpárt az albérletbe vagy a kollégiumba, de ennek ellenére kerékpárral akarnak közlekedni Debrecen városában.

Az általam szemléltetett szolgáltatás legfőbb jellemzője, hogy több telephellyel rendelkezik, amelyek ugyanahhoz a tulajdonoshoz kapcsolódnak. Ezen telephelyek a következő egyetemi campusokon lennének elhelyezve:

- Debreceni Egyetem főépület
- Böszörményi úti campus
- Kassai úti campus
- DE-OEC
- Ótemető utcai campus

A több telephelyből adódóan a kölcsönző lehetőséget nyújt arra, hogy a kerékpárt nem azon a telephelyen adjuk le, mint amelyiken felvettük így kitűnően alkalmas lehet azon emberek

részére, akik a város egyik pontjából szeretnének eljutni egy másikba. A szolgáltatás egyik célcsoportját azon hallgatók adják, akiknek a képzésükből adódóan a kurzusaik nem egy egyetemi campuson találhatóak, hanem a Debreceni Egyetem különböző campusain hallgatnak előadásokat. Ezen kurzusok nem ritkán egymás után következnek, tehát az utazásra a campusok között nagyon kevés idő áll rendelkezésére a hallgatóknak. A Debreceni tömegközlekedés lehetőséget nyújt bizonyos járatok használatára, melyekkel megoldható az utazás a campusok között, de ezek a járatok nem a hallgatókhoz és nem az előadásokhoz vannak igazítva, így van olyan eset, hogy a rendelkezésre álló idő alatt nem lehet tömegközlekedéssel elérni a következő előadástt. További célcsoport vonható be a cég vonzáskörzetébe esetleges új telephelyek létesítésével, melyek a „panelvárosok" mellett kerülnének elhelyezésre. Ezek a telephelyek igény és tetszés szerint változtathatóak, bővíthetőek. Ezzel a terjeszkedéssel nem csak az onnan bejáró diákokat lehetne bevonni a szolgáltatás használatába, de az egyre bővülő kerékpárút infrastruktúrával a Debreceni lakosság jelentős hányada is potenciális felhasználóvá válna. Ezen bővítési lehetőség szerves részét képezi az általam tervezett nyilvántartási rendszernek, ezzel segítve a szolgáltatás teljes körű testreszabhatóságát. A szakdolgozatnak nem része a kölcsönző teljes megtervezése, mindössze egy esetleges opciót használ a nyilvántartó rendszer bemutatására.

A kerékpárkölcsönző működése a következőképp épül föl. A szolgáltatás egy könyvtár jellegű kölcsönzési rendszerrel rendelkezik, tehát nem önkiszolgáló és nem automatizált, szükséges egy eladó jelenléte. A telephelyek száma és elhelyezkedése a program szempontjából nem korlátozott, ezek elhelyezése a nyilvántartó rendszer szempontjából irreleváns. Ezen telephelyek egy fedett területen helyezkednek el, ahol lehetőség van a kerékpárok elzárt tárolására. A telephelynek rendelkeznie kell egy internetelérással rendelkező megfelelő számítógéppel, melyen egy egyénileg választható böngésző futatható. A nyitvatartási időben a telephelyen kell tartózkodnia egy kiszolgáló személynek, aki az ügyfelekkel foglalkozik. Az ő munkakörébe tartozik a nyilvántartó rendszer kezelése, mely a már említett számítógépen történik. A kiszolgáló személy kötelessége a rendszerbe bejegyezni minden kerékpármozgást, mely érinti az általa felügyelt telephelyet. Továbbá a kerékpárok tényleges fizikai kiadása és visszavétele is az ő hatáskörébe tartozik.

3.2 A megrendelő igényei

A nyilvántartó rendszer kifejezetten a már fentebb vázolt kerékpárkölcsönző kezelésére készült, de a felhasznált technológiáknak köszönhetően tetszés szerint bővíthető. Ezzel lehetőséget adva a kerékpárkölcsönző esetleges tényleges elindítása esetén a program az adott megoldásra történő specifikálására.

A Debreceni Egyetem, mint megrendelő egy kerékpárkölcsönzőt fog beindítani és ehhez keres nyilvántartó rendszert. A kölcsönző kialakításából adódóan a legfontosabb szempont, ami alapján a rendszert meg kell tervezni, az a több telephely. A programnak képesnek kell lenni több különálló telephelyet kezelni. A telephelyek a működés során nem különálló kerékpárállománnyal rendelkeznek, hanem közös állománnyal dolgoznak, így a legfontosabb, hogy az adatokat nem lehet telephelyenként külön tárolni és kezelni. Az időben eltolódott szinkronizáció sem lehetséges, mivel az inkonzisztens állapotot hozhat létre a különálló adatbázisokban. Ebből adódóan a legmegfelelőbb és legköltséghatékonyabb megoldás egy weboldalba építeni a programot, amelyet mindegyik telephelyről el tudnak érni, egyidejűleg mégis elkülönítve. Az adatokat egy közös adatbázisban tároljuk, ezzel biztosítva, hogy a különálló telephelyek mégis közös adatokkal dolgozzanak. A telephelyek közötti kapcsolat kiépítésére nincs szükség, hiszen a legköltséghatékonyabb megoldás az internet használata, a telephelyek hálózatba történő kapcsolására.

A megrendelő szeretné online foglalás lehetőségét biztosítani a már regisztrált tagoknak, ezzel növelve a felhasználók kényelmét és így elnyerni a bizalmukat és minél több a szolgáltatást igénybe vevőt a regisztrációra ösztönözni. Így a regisztrált tagok adataiból könnyebb statisztikát készíteni, hogyan vették igénybe a szolgáltatást, melyek azon pontjai a cégnek melyek fejlesztésre, átalakításra szorulnak, melyek azok, amelyeket teljesen el kell törölni vagy alapjaitól átalakítani. Ezen statisztikák irányt adhatnak a cégnek, a fejlesztések, esetleges bővítések felé, melyek segítségével ezek a beruházások sikeresebbek lesznek, így növelve a cég profitját. Az online foglalás lehetőségét csak a regisztrált tagok vehetik igénybe, mivel az internet anonimitása és az esetleges rosszindulatú felhasználók okozta károk így a legegyszerűbben kiküszöbölhetőek. A foglalás lehetőségét legegyszerűbben a bemutatkozó weboldal részeként tudjuk a felhasználók használatára bocsátani. A weboldalban létrehozunk egy bejelentkezés menüpontot, melyben a felhasználók a már regisztrált adataikkal beléphetnek, ezzel azonosítva magukat. A bejelentkezés a foglalási oldalhoz vezeti a

felhasználót ahol megadhatja a foglalás részleteit, mely telephelyen és mikor kívánja felvenni a választott kerékpárokat.

Bizalmi szolgáltatás lévén, a megrendelő nem szeretné megnyitni a szabad regisztráció lehetőségét a nagyközönség számára. Ez az előbbiekben vázolt rosszindulatú internetes felhasználók okára vezethető vissza, akik egy álneves regisztrációval könnyen hozzáférhetnének a cég tulajdonát képező kerékpárokhoz és nem lenne lehetőség az igazi személyazonosságuk azonosítására, így nem lehetséges a fellelőségre vonásuk sem, egy a cég tulajdonában történő károkozásért. A károkozás keretébe akár a teljes kerékpár eltulajdonítása is megtörténhet, ezzel jelentős anyagi kárt okozva az azt működtető cégnek. A rendszerbe történő regisztrációt egy belépési nyilatkozat kitöltéséhez köti a megrendelő, melyet bármelyik telephelyen megtehet a jelentkező. A nyilatkozat kitöltése és az esetleges tagdíj befizetése valamint a személyes okiratok bemutatása után az eladó képes meggyőződni a regisztrálandó vásárló személyazonosságáról, így a regisztrációnál nem vagy csak nagyon nehezen lehet hamis adatokkal taggá válni. Az eladó az erre a részre kialakított, csak a telephelyen dolgozók által hozzáférhető felületen regisztrálja az új tagot és kiállítja számára a tagsági kártyát, mellyen szerepel a rendszer által kiállított tagsági szám. Ezzel az azonosítószámmal könnyen azonosíthatja magát a kölcsönzésnél a már tagságot kiváltott vásárló. A nem regisztrált tagok kölcsönzése is egy bizonyos regisztrációhoz kapcsolódik, melyet a felhasználó első kölcsönzése során az eladó szintén köteles a rendszerben feltüntetni. Ezen regisztráció nem jár tagsággal, mindössze a legfontosabb személyes adatait rögzíti így a rendszerben a kölcsönzőnek, mivel így lehetséges bármilyen jellegű számonkérése az illető személynek a kölcsönzéssel kapcsolatban. Legyen szó késői leadásról, megrongált alkatrészről vagy vissza nem jutatott kerékpárról. Így látható, hogy a kölcsönző bizalmi szolgáltatás lévén, köteles a felhasználók adatait tárolni és kezelni, függetlenül a szolgáltatáshoz kötődő tagságukra nézve, azaz nincs lehetőség anonimként igénybe venni a kerékpárkölcsönzői szolgáltatást.

A megrendelő igényeit figyelembe véve egy adatbázis háttérrel rendelkező két belépési ponttal bíró weboldalcsoportot hoztam létre. A bemutatkozó weboldal marketing célokból is létrejött volna, de az online foglalásra történő igény miatt ez az opció is ide lett beépítve. Ez egy login rendszerrel lett lezárva a nem regisztrált felhasználók elől. Az telephelyen dolgozók számára készített nyilvántartó rész négy fő részből áll. Ezen részek egymástól elkülönülő funkciókat tartalmaznak, de azonos adatbázison dolgozik mindegyik, így nem lehet teljesen különállónak tekinteni őket, egy rendszer részei. Az eladónak is azonosításra van szüksége a

munkamenet legelején, tehát a funkciók nem elérhetőek a bejelentkezés előtt. A belépés után választhatunk a négy funkció közül, melyikre van szükségünk jelenleg. Ezen funkciók a már említett regisztráció, ahol a leendő tagok regisztrációja történik. A kerékpárok kiadása és visszavétele külön funkció, melyben a kiadásnál elindítunk egy kölcsönzést, melyhez rendelni kell egy személyt és kölcsönözni kívánt kerékpárokat a megfelelő időbélyeggel. A visszavétel során a már megnyitott kölcsönzéseket tudjuk lezárni, de figyelembe kell venni, hogy nem feltétlen azonos telephelyen történik a lezárása a kölcsönzésnek, mint a megnyitása, így az összes telephelyen megnyitott kölcsönzést kezelni kell tudnunk az összes telephelyen. A negyedik opció a statisztikák elkészítésénél és a forgalom áttekintésénél jut fontos szerephez. Ez a funkció az áttekintés, mely lehetőséget nyújt az adatbázis megfelelő adatainak megtekintésére, lekérdezésére így készítetünk kimutatásokat például az aznapi kölcsönzésekről. Ezen funkciók összessége adja nyilvántartórendszer felhasználói részét, melyet a telephelyen dolgozók kezelnek.

3.3 Rendelkezésre álló erőforrások

A Debreceni Egyetem célja egy olyan rendszer kialakítása, melynek beüzemeléséhez és tartós fenntartásához nincs szükség jelentős beruházásra technológiai téren. Azaz a már kiépített informatikai rendszerek segítségével jól működtethető. A nyugati modellű, automata kerékpárkölcsönzők melyek például Párizs városában láthatók, jelentős kezdőbefektetést igényelnek, az általam választott kölcsönző rendszerrel szemben. Ezen rendszerek kiépítéséhez jelentős infrastruktúrás fejlesztések szükségesek.

A jelen esetben kiválasztott rendszer esetében egy elkülönített állomás létrehozása szükséges a campusok területén, melyben egy interneteléréssel rendelkező számítógéppel és egy alkalmazott működőképes kerékpárkölcsönzőt működtethet. Ennek a rendszernek a kiépítéséhez, a befektetés összege jelentősen alacsonyabb, mint az előző automata rendszerhez szükséges összeg.

Az általam készített nyilvántartó rendszer szempontjából releváns erőforrásokat és a megrendelő igényeit figyelembe véve, a program legoptimálisabb implementációja egy közös adatbázisrendszert használó weboldalak gyűjteménye. Melynek készítéséhez freeware programokat használtam. A működtetés megoldható a már üzemelő egyetemi szerverek segítségével, tehát nincs szükség további beruházásokra, az informatikai rendszer

fejlesztésére. A telephelyek hálózatba kapcsolására, az egyetemi internetes hálózat a legcélszerűbb és legköltséghatékonyabb megoldás. Ezen opciókat használva egy az erőforrásokat jól kihasználó, könnyen fenntartható kerékpárkölcsönzőt kapunk.

4. Program bemutatása

4.1 Adatbázis felépítése

Az adatbázis kezelésére az ingyenes phpMyAdmin szoftvert használtam, melyet PHP nyelven készítettek és a MySQL adatbázisok adminisztrálására szolgál. Egy egyszerű böngészőből elérhető szoftverről van szó, melynek segítségével grafikus felületen kezelhetjük a MySQL adatbázisunkat. A nyilvántartórendszer adatbázisának karbantartására is a phpMyAdmin szoftvert ajánlom.

admin

Mező	Típus
id	tinyint(4)
username	varchar(65)
password	varchar(65)
telep	tinyint(4)

members

Mező	Típus
id	int(4)
username	varchar(65)
password	varchar(65)

kolcson

Mező	Típus
id	int(11)
tag	tinyint(4)
ki	int(11)
melyiket	varchar(300)
mikortol	timestamp
mikorig	timestamp
honnan	tinyint(4)
hova	tinyint(4)
allapot	tinyint(4)

elado

Mező	Típus
id	tinyint(4)
nev	varchar(200)
szuldatum	date
lakcim	varchar(300)
tel	varchar(20)
email	varchar(100)
telep	tinyint(4)

tagok

Mező	Típus
id	tinyint(4)
nev	varchar(200)
szuldatum	date
lakcim	varchar(300)
email	varchar(100)
tel	varchar(20)

kerekpar

Mező	Típus
id	tinyint(4)
sorszam	int(11)
allapot	tinyint(4)

ideiglenes

Mező	Típus
id	tinyint(4)
nev	varchar(200)
szuldatum	date
lakcim	varchar(300)
tel	varchar(20)

foglalas

Mező	Típus
id	tinyint(4)
ki	tinyint(4)
mennyit	tinyint(4)
hol	tinyint(4)
mikor	varchar(300)
allapot	tinyint(4)

telephely

Mező	Típus
id	tinyint(4)
nev	varchar(200)
cim	varchar(300)

A fenti kép a phpMyAdmin szoftverből készült, melyen az adatbázis struktúráját mutatom be. Helyhiány miatt a struktúrának csak a lényeges része látható és nem az eredeti megjelenésben. A vastaggal szedett nevek a táblák neveit takarják, ezek alatt találhatóak a táblában lévő mezők és azok típusai. Minden tábla rendelkezik egy id azonosítóval, amely elsődleges kulcsként működik és, melynek értéke mindenkor az úgy nevezett auto_increment. Ez az opció minden új sornál eggyel növeli a számlálót, mely a pozitív egész számokat reprezentálja. Ezzel a kulccsal lehet összekapcsolni a tagok-members és az elado-admin táblákat. Más táblák esetén is használom az id-t, például a telephely tábla id mezőjének értékét használom fel a kölcsön tábla honnan és hová mezőinél, ahol a telephelyeket kell tárolnom, de nem azok neveivel kívánok dolgozni az adatbázison belül mindössze az azonosítókkal. Ha felhasználni kívánom ezen értékeket, mindössze visszakeresem az azonosítószámhoz milyen nevű telephely kapcsolódik és azt iratom ki, nem pedig az azonosítót magát.

Az adatbázisban többnyire stringeket tárolok, de ezek mellett szerepel még a timestamp típus. Ezen típus az időbélyeget jelenti, melyet a kölcsönzés megkezdése és leadása esetében említettem. A timestamp egy olyan típus, amely az év-hónap-nap óra:perc:másodperc formátumban tárolja az időt. A kölcsönzés indításánál nem szükséges megadom az időt, mivel egy új sor létrehozásakor az alapértelmezett beállítása ennek a mezőnek a CURRENT_TIMESTAMP, így az aktuális időt rendeli egyből a mezőhöz. A login rendszernél tárgyalt md5() titkosítással tárolt jelszavak is egyszerű varchar() típusú mezőkben vannak tárolva, mivel az md5() függvény eredményét nem lehet visszafejteni, így azt egyszerű stringként lehet tárolni.

Az adatbázisról megfelelő időközönként biztonsági mentést kell készíteni, ezt megoldható a phpMyAdmin export funkciójával. Ezzel biztosítani lehet egy esetleges adatbázis hiba esetén egy régebbi, de ép állapot visszaállítását. Az adatbázisban történő hibák minimalizálása végett a nyilvántartó rendszerben szereplő össze adatbázis művelet megőrzi az adatbázis egységes állapotát, nem jöhet létre olyan állapot, melyben sérülne az adatbázis. Ezzel biztosítva a kerékpárkölcsönző számára készült nyilvántartó rendszer zavartalan, helyes működését. Ha mégis sérülne az adatbázis, akkor az adminisztrációs felület segítségével könnyen kijavítható a hiba, vagy visszaállítható egy régebbi állapot a biztonsági mentések segítségével.

4.2 Bemutató weboldal

4.2.1 Weboldal és kerete

A nyilvántartó rendszerhez kapcsolódó weboldal két fő funkciót tölt be a Debreceni Egyetem igényeihez igazodva. Az elsődleges funkciója a szolgáltatás bemutatása, leírása egy egyszerű weboldal segítségével, melyet bárki elérhet. Ez a weboldal a marketing szerepén felül információközlőként is funkcionál. Ezen a weboldalon tájékozódhatnak a régi felhasználók és a potenciális új vásárlók a szolgáltatás lényeges paramétereiről, többek közt a telephelyek elhelyezkedéséről, valamint az aktuális szolgáltatási díjakról. A weboldalba épített aktív funkció a már bemutatott online foglalási rendszer, melyet a felhasználók igényeihez igazítva alakítottam ki, ezzel elősegítve a szolgáltatás igénybevételét.

A weboldal implementációja jelenleg a minimál dizájn szemléletében készült, melynek gyakorlati előnye az esetleges tényleges felhasználásra kerüléskor érzékelhető. A program valós felhasználása esetében, az adott időszak és a kiválasztott cégprofilnak megfelelő megjelenéssel ruházható fel a weboldal felhasználók számára készült része. A weboldal felépítésének kialakításánál szem előtt tartottam azon szempontot, hogy a tényleges üzembe helyezésnél minimális erőforrás befektetéssel lehessen kialakítani a végleges weboldalt.

A weboldalt a napjainkban elterjedt 1024x768 felbontású megjelenítő eszközökre optimalizáltam, de az egyre terjedő 16:9-es megjelenítési arányú eszközökön is megfelelően jelenik meg. További fejlesztésre adhat okot a hazánkban is egyre népszerűbbé váló okos telefonok és az ezen eszközöket internetelérésre használó felhasználók terjedése. Melyek következtében kilátásba helyezhető egy mobiltelefonokra optimalizált változata a weboldalnak.

A weboldal felépítése a következőképpen történik:

- A kezdőoldal teljes szélességében a felső 20%-ában az oldalnak helyezkednek el a logók. Két logó kapott helyett a jelenlegi implementációban, melyek egymás mellett helyezkednek el. Az első a menü szélességében készült és a Debreceni Egyetem logójának van fenntartva ez a hely. A következő a weboldal fennmaradó szélességében található és a kerékpárkölcsönző saját logójának helyét foglalja le a weboldalból. Jelenleg helyfoglaló jelleggel szerepelnek képek az adott helyeken, a

tényleges üzembe helyezés alkalmával ezek könnyedén kicserélhetőek a megfelelő logókra.

- A weboldal bal oldalán helyezkedik el a menü. A már említett szélességben, mely megegyezik az első logóéval. A menü teljes tartalmát a már bemutatott „include” segítségével jelenítem meg. A kezdőlapon mindössze a menü számára fenntartott hely szerepel a forráskódban kiegészítve a megfelelő PHP utasítással, mely a „<?php include 'menu.php' ?>”. Ezen utasítás a menu.php fájl teljes tartalmát elhelyezi a kezdőoldal menü számára fenntartott helyén. A menu.php tartalma a kezdőlaptól függetlenül változtatható, így megkönnyítve a menüpontok szerkeszthetőségét.
- A kezdőlap fennmaradó központi helyén a munkaterület található. Ezen a területen jelenik meg az általunk kiválasztott menüpontnak megfelelő információ. Ezen területre is az include segítségével jelenítjük meg a tartalmat. A meghívott fájl készítésénél az a terület ahova meghívjuk a teljes munkafelülete az adott fájlnak, így könnyen szerkeszthető a felépítése az oldalunknak. Létrehozzuk a felépítést a kezdőoldalon és a megfelelő tartalmakat include segítésével meghívjuk, melyeket könnyen lehet formázni, mivel csak a meghívott területet fogják használni.

A weboldal kerete ezen bemutatott séma szerint készült. A felhasználó, ha meglátogatja az oldalunkat, az „index.php”-ra fog megérkezni, mely mindössze az oldalunk szerkezeti felépítését tartalmazza. A szerkezeti elemekbe beépített meghívások segítségével töltjük fel az oldalt tartalommal. A menu.php fájl az általunk választott menüpontokat tartalmazza, melyek táblázatba szedve szerepelnek és minden menüpont egy speciális linket tartalmaz, amely a következőképpen épül fel „Áraink”. A példa az áraink menüpontot reprezentálja, a szokásos HTML kifejezések segítségével. A link nem visz el a kezdőoldalról, mindössze az url-ben elhelyez egy menu változót az aktuális értékkel, mely minden menüpont esetében egyedi. A kezdőlapunk központi munkafelülete ismételten egy üres rész a forráskódban, melybe a nav.php fájlt hívjuk meg. A navigációs célokat szolgáló fájl feladata a megfelelő tartalmi oldal meghívása a kezdőoldalba. Minden menüponthoz tartozik egy tartalmi oldal, melyben a menüpontnak megfelelő információkat tároljuk. Ezzel a felépítéssel könnyen létrehozhatunk, törölhetünk vagy módosíthatunk bármely menüpontot vagy azok tartalmát. A navigációs fájl lekérdezi a menu változó értékét az url-ből és a változónak megfelelően a kiválasztott tartalmi fájlt hívja meg. Az oldal első letöltésekor, mikor még nincs értéke a változónak az alapértelmezett kezdőoldali fájlt hívja meg. A jelenlegi implementációba a következő menüpontokat építettem be:

- Főoldal
- Magunkról
- Áraink
- Útvonalak
- Galéria
- Kapcsolat
- Partnereink
- Foglalás

Ezen menüpontok segítségével egy átfogó, jól szerkesztett bemutatkozó weboldalt sikerült létrehoznom. Amelyet a bemutatott include használatának köszönhetően, könnyen át lehet alakítani a mindenkori igényeknek megfelelően.

4.2.2 Foglalás

A weboldal foglalás funkciója könnyen elérhető kell, hogy legyen és az átlagfelhasználó számára is elsőre könnyen kezelhető felületet kell, hogy biztosítson, ezzel elősegítve a szolgáltatás igénybevételét és a minél széleskörűbb elterjedését.

A könnyű elérhetőség biztosított, mivel a foglalás menüpont szerepel a kezdőlap menüsorában, így nem kell külön keresgélnie a felhasználónak. A menüpontra kattintva a már bemutatott munkafelületen egy bejelentkezési ablak tűnik fel. A szolgáltatásba való belépést már bemutattam az előző fejezetekben, összegezve: személyesen kell regisztrálni a telephelyek egyikén, így általunk megadott felhasználói névre és generált jelszóra teszünk szert. A login rendszer bemutatása a következő fejezetben történik. A belépési ablakban helyesen megadott felhasználói név és jelszó páros megadása után megjelenik a foglalási oldal, mely szerkezetében megegyezik a bemutató weboldallal. Különbség a menü megváltozása, melyben a szükséges menüpontok megmaradnak, jelenleg ezek a foglalás és áraink, melyek kibővülnek egy kijelentkezés opcióval is. A kezdőlapon alapesetben a foglalási kérdőívet magába foglaló tartalmi fájl hívódik meg. A foglalási kérdőív a mindössze feltétlenül szükséges paramétereket tartalmazza, így gyorsítva és egyszerűsítve a foglalási procedúrát. A foglalás egy egyszerű HTML űrlap kitöltésével történik, melyben legördülő opciók közül tudjuk a számunkra megfelelőt kiválasztani. Ezzel a megoldással lehetőséget nyújtunk a felhasználónak a foglalás széleskörű specifikálására, de kiküszöböljük az elgépelés vagy hibás adat megadásának lehetőségét.

A bekért adatok a következőek: a kölcsönözni kívánt kerékpárok száma, azon telephely megnevezése, melyen felvenni kívánja a kerékpárokat és a kívánt időpont, mikor felvenni kívánja azokat. A kerékpárok száma és a kívánt telephely egy-egy legördülő menüből kiválasztható. Az időpont megadása a hónap, nap, óra és perc külön-külön való megadásával történik. Ezen adatokkal egyértelműen rögzíthető a foglalási szándéka a felhasználónak. A „Lefoglalom” gombra való kattintás után a „sum.php” oldalra irányítódik át a felhasználó, melyen megtekintheti a foglalásának az adatait összegezve és értesítve lesz a foglalás sikeres regisztrációjáról vagy az esetleges hibáról. Az összegzés után található egy „Vissza” feliratú link, mely segítségével visszatérhet a kezdőoldalra és ott a kijelentkezés segítségével befejezheti az adott munkamenetet.

Az összegzési oldal megjelenésekor a forrásban található PHP kód úgy fut le, hogy a felhasználó már csak az eredményképpen kapott kimenetről értesül. A foglalási kérdőívben megadott adatokat beolvassa és elmenti egy-egy változóban, ez a „$_POST['változó']” paranccsal történik, a változó helyére az űrlapban megadott aktuális nevet adjuk meg ezzel kiválasztva, mely elem eredményét kérjük le. Ezen utasítást egy változó értékéül adva megkapjuk az eredményét az űrlap megfelelő elemének. Az adatbázishoz kapcsolódás után megkeressük a bejelentkezett személy adatait a felhasználói név alapján. Létrehozunk egy $sql változót, melyben elhelyezzük azt az SQL utasítást, mellyel a foglalás táblába létrehozunk egy új sort a kapott adatokkal, az utasítás így néz ki:

```
$sql="INSERT INTO $tbl_name (ki, mennyit, hol, mikor, allapot)
VALUES ('$tag', '$szam', '$hely', '$honap-$nap-$ora:$perc', '0')";
```

Ezt a mysql_query($sql) paranccsal lefuttatjuk és figyelve az eredményét a sikeres foglalást és az adatok összegzését írjuk ki a kimenetre vagy a megfelelő hibaüzenetet. Ha sikeresen lefut ez az utasítás, akkor a foglalás regisztrálása megtörtént a rendszerben.

4.2.3 Login rendszer

A nyilvántartó rendszer fontos részét képezi a tagok számára fenntartott online foglalási rendszer, melyet a már regisztrált tagok vehetnek igénybe, de lényeges a nyilvántartási rendszer védelme a nem jogosult felhasználókkal szemben. A weboldal ezen részeinek

védelmét a login rendszer biztosítja, mely csak az arra jogosult felhasználóknak enged hozzáférést a védett tartalmakhoz.

A foglalási rendszer elérhető a bemutató weboldalról, ezzel szemben a nyilvántartó rendszer címét nem hozzuk nyilvánosságra, de nyílt hálózaton szerepel, tehát elérhető mindenki számára, ezért szükséges a használók azonosítása. A két védett részt felhasználók külön csoportból kerülnek ki, ezért külön login rendszerrel kellett ellátnom a foglalási és a nyilvántartási részt, de a két rendszer algoritmusa megegyezik. Az azonosítási folyamat egy felhasználói név és jelszó páros megadásával történik. Az adatbázisban külön táblában tárolom a felhasználók és a kölcsönzőben dolgozók adatait, így a két login rész külön táblákkal dolgozik. Az adatbázisban tárolt név szabadon olvasható, de a jelszót minden esetben a MySQL beépített md5() titkosító függvényével titkosítom tárolás előtt.

A belépéshez szükséges adatok megadása egy HTML űrlapon történik, ahol a nevet egyszerű szövegként, a jelszót „password” típusként olvassa be a program, így azt a gépelés során sem lehet elolvasni. A jelszót azonnal md5() függvénnyel titkosítom, így sehol nem szerepel egyszerű szövegként letárolva. Az űrlap egy ellenőrző oldalra irányul, ahol csatlakozom az adatbázishoz és a megfelelő táblában lefuttatok egy keresést a kapott adatokra. Az eredménytábla vizsgálatánál a sorok számát ellenőrzöm, ha eggyel megegyezik, akkor van találat tehát az adott felhasználói név és jelszó megtalálható az adatbázisunkban, tehát jogosult a belépésre a személy. Ekkor a PHP SESSION változóját veszem igénybe, amellyel egy munkamenetet indítok el, a „session_register("myusername");” paranccsal. A program kapcsolja még a titkosított jelszót egy következő változóban így létrehozva a munkamenetet, amely hitelesíti az adott bejelentkezést a kijelentkezés gomb megnyomásáig vagy a böngésző bezárásáig. A védett oldalak létrehozásához arra van szükség, hogy az oldal forráskódjában bármely más utasítás előtt vizsgáljuk a session_is_registered(myusername) változót, hogy létezik-e. Az utasítás igaz vagy hamis értékkel tér vissza, ezt vizsgáljuk, és ha hamis, akkor egy hibaüzenet kíséretében visszairányítjuk a látogatót a login képernyőre. Ha igaz értékkel tér vissza, abban az esetben lefut az oldalon található többi utasítás így hozzáférést biztosítva a védett oldalhoz. A session változót több helyen is igénybe veszem a programom folyamán, leginkább azon helyeken, ahol a belépett személy azonosságát kell felhasználni egy-egy folyamatban, mint például a rendelés esetében ebből a változóból keresem, vissza kihez kell hozzárendelni a foglalást.

A nyilvántartó rendszer login része teljesen megegyező algoritmus szerint épül fel. A lényeges különbség a session változók regisztrálásában rejlik, mivel a nyilvántartó rendszernél több ilyen változót vizsgálok, ezzel ellenőrizve a bejelentkezést. A korai szakaszában a fejlesztésnek egy olyan problémába ütköztem, hogy azonos változókat regisztráltam mindkét esetben és azonos eseteket ellenőriztem mind a foglalási mind a nyilvántartó rész védett oldalainál. A login ellenőrzése esetében nem az egyezését vizsgálja az algoritmus az adatoknak, mindössze azt, hogy létezik-e a megfelelő session változó. Ekkor fordult elő az a hiba, hogy ugyanabban a böngészőben a foglalási részbe bejelentkezve a felhasználó elérhette a nyilvántartó rendszer védett oldalait, ami jelent esetben minden részére kiterjed a weblapnak. Ezt a problémát egy csak a nyilvántartó rendszert kezelő személyekhez rendelt változó session regisztrálásával értem el. Ezt a változót hozzávettem a jogosultságellenőrzéshez a megfelelő oldalak forrásában, így nem lehetséges, hogy a különböző helyen belépett felhasználók a rendszer más részeit elérjék, az ott történő azonosítás nélkül.

4.3 Nyilvántartórendszer

A weboldal azon része, melynek címét nem hozzuk nyilvánosságra, a kezeléséért a telephelyeken dolgozó személyzet a felelős és a következő funkciókat látja el: a kerékpárkölcsönző szolgáltatásait igénybevevő személyek adatainak a tárolása, a foglalások regisztrálására, a kerékpárok kiadásának és visszavételének rögzítése, azaz a kölcsönzések regisztrálása egy jól kezelhető, átlátható rendszerben, valamint a rendszerben történő különféle kereséseket valósítja meg.

A kezdőoldal a bejelentkezési ablakkal kezdődik, ahol az eladóknak külön-külön generált felhasználói név és jelszó párossal azonosítani kell magát. A sikeres azonosítás után a kezdőoldalon a négy fő funkciót reprezentáló kép helyezkedik el a képernyő teljes felületén. Ezen képek linkként funkcionálnak és a megfelelő oldalra irányítják a felhasználót, a kiválasztást követően. Az ezt követő munkafelület bármely esetben azonos felosztású az elvégzendő feladatok gyorsítása végett. A bal oldali sávon helyezkedik el egy menüsor, melyből elérhetjük a kezdőképernyőt, a négy fő funkcióoldalt és egy kijelentkezés gomb segítségével befejezhetjük az adott munkamenetet. A képernyő fennmaradó felületét az aktuális munkafelület foglalja el, melyen a kiválasztott funkciónak megfelelő tartalom jelenik

meg. Ezen a felületen lehet a kiválasztott feladatokat elvégezni, itt jelennek meg az adatok bevitelére alkalmas mezők és erre a felületre érkeznek meg a lefutatott parancsok eredményei is. A nyilvántartó rendszer fő funkciói mellett, azok kiválasztása után lehetőségünk van választani további alfunkciók közül, melyek tovább specifikálják az elvégzendő feladatot. A további alfunkciók a munkafelület fölött jelennek meg, egy vízszintes menüsorhoz hasonlóan. Ezzel a kialakítással gyorsan kezelhetővé és könnyen átláthatóvá válnak a rendszert működtető funkciók már az első használat alkalmával is. A több telephellyel rendelkező kerékpárkölcsönzőt érintő kerékpármozgást és a szolgáltatást igénybevevő személyek, legyenek regisztrált tagok vagy egyszeri vásárlók, adatait a kialakított nyilvántartó rendszer gyorsan és hatékonyan képes kezelni, tárolni és többféle lehetőség szerint visszakeresni.

4.3.1 Regisztráció

A rendszerbe történő regisztráció egy fontos funkciója a nyilvántartó rendszernek, mivel a kerékpárkölcsönző lehetőséget biztosít az azt igénybe vevő személyeknek, hogy kiváltsák a tagságukat, így különféle előnyökhöz jussanak. A regisztráció a telephelyeken történik, a személyes megjelenés kötelező. A nyújtott szolgáltatás jellege miatt, a kerékpárkölcsönzés nem jöhet létre az adatok megadása nélkül. Ezért nem csak a belépett tagok adatainak tárolás és kezelése szükséges, hanem igény van az egyszeri, továbbiakban ideiglenes, vásárlók adatainak tárolására. Ezért a regisztrációs menüpontban található további két lehetőség, melyeket kiválasztva tovább specifikálhatjuk, milyen regisztrációt szeretnénk folytatni.

Az egyik lehetőség, mely rendelkezésünkre áll, az ideiglenes tag regisztrálása. Ebben az esetben a szolgáltatást igénybe vevő személy nem akar csatlakozni a kerékpárkölcsönző tagsági rendszeréhez, mindössze alkalmi kölcsönzést kíván végrehajtani. Ebben az esetben a telephelyen munkában lévő személy egy a személyazonosság igazolására szolgáló okirat megtekintése után a munkafelületen található HTML űrlapot kitöltve felveszi a vásárló főbb adatait, melyek segítségével azonosítani és kapcsolatot teremteni lehet az illetővel. Ezek az adatok a név és születési idő, melyek segítségével azonosítani lehet a személyt, továbbá a lakcím és a telefonszám, melyen kapcsolatot lehet létesíteni vele. Ezen adatok bevitele gyors és egyszerű, így elősegítve a kölcsönző működését. A regisztrál gomb megnyomásával az adatok a már bemutatott módon átadódnak a következő php oldalnak, amely ellenőrzi, hogy minden mezőt kitöltöttek-e, ha igen, akkor csatlakozik az adatbázishoz és az ideiglenes nevű

táblába hozzáad egy új sort a kapott adatokkal, így létrehozva egy új felhasználót. A „mysql_insert_id()” függvényt használva és értékét egy változóban tárolva újabb lekérdezés nélkül megkaphatjuk, az új sor id értékét, mely a jelen esetben a felhasználó ideiglenes azonosítója, melyre szükség van a kölcsönzés során. Ezt az értéket az eredményoldalon kiíratom, így egy új vásárló felvétele után azonnal tudható az azonosítója, mellyel a kölcsönzés megkezdhető.

Ha egy vásárló a tagság előnyeit kihasználandó, ki szeretné váltani a tagságot, akkor a tag regisztrációt kiválasztva tehetjük meg, hogy regisztráljuk a rendszerbe. A tagsági regisztráció nagyban hasonlít az ideiglenes vásárló regisztrációjához, de pár lényeges dologban eltér. Ebben az esetben is egy HTML űrlap kitöltésével kezdődik a folyamat, melyen az eddig említett adatokon kívül megtalálható még az e-mail cím és a felhasználói név mező is. Ezen adatok az online foglalási rendszer használatához szükségesek. A regisztráció algoritmusa megegyezik az előzőekben vázolttal, de a jelen esetben a tag táblát használjuk az adatok tárolására. A különbség, hogy minden taghoz tartozik egy újabb sor a members nevezetű táblában, ahol a felhasználói nevet és jelszót tároljuk. A jelszót biztonsági okokból úgy döntöttem, hogy nem a vásárló adja meg, mivel így harmadik személy, jelenleg az eladó, is tudomást szerezhet róla, többek közt írásos bizonyíték is készülne ebben az esetben a jelentkezési lap kitöltésekor. Ezen megfontolásból egy generált jelszó mellett döntöttem, melyet a megadott e-mail címre elküldve, a megadott e-mail cím megerősítése is megtörténik. A jelszó generálása egy előre megadott tömbből történik, melyben az angol abc némi kihagyással és a pozitív egész számok nullától kilencig szerepelnek. Ebből a tömbből véletlenszerűen generálok egy előre beállított hosszúságú jelszót, mely jelen esetben hetes értékre van beállítva. A személyes adatok a tag táblában lesznek tárolva, melynek id értéke megegyezik a members tábláéval, melybe a foglalási rendszerbe való belépéshez szükséges adatok lesznek tárolva. A jelszó természetesen md5() titkosítással szerepel a táblában, melyet a login rendszer fejezetben már kifejtettem. A megadott e-mail címre egy üdvözlő üzenet keretein belül a felhasználó hozzájut a számára generált jelszóhoz. Ezzel a módszerrel egy esetleges külső személy nem juthat hozzá a felhasználói névhez és jelszóhoz egyszerre, mivel azokat külön időben és helyen jutnak a vásárló tudomására. Az eredményoldalon hiba esetén a megfelelő hibaüzenet, sikeres regisztráció esetén pedig a sikerült üzenet mellett az újonnan regisztrált tag tagsági azonosítóját iratom ki, így azt felhasználva azonnal lehetséges kölcsönzést indítani.

4.3.2 Kölcsönzés megkezdése

A kölcsönzés megkezdéséhez szükséges a kölcsönözni kívánó személy regisztrálása a rendszerbe az előző fejezetben bemutatott módon. Ha az illető már tag vagy ideiglenes vásárlóként szerepel a rendszerben, akkor nincs további teendő, mint a kiadás menüpontot kiválasztva megkezdjük a kölcsönzési folyamatot.

A kiadás menüpontot kiválasztva egy HTML űrlap jelenik meg, melyben az első elem egy úgy nevezett radio gomb. Ez az elem a HTML-nek egy olyan eszköze mellyel több lehetőség közül tudjuk kiválasztani a nekünk és a helyzetnek megfelelőt. A jelen program esetében két lehetőséget kellett reprezentálnom. A két lehetőség a kölcsönözni kívánó személy státuszára utal, mely lehet tag vagy ideiglenes vásárló. Ezt a két lehetőséget egymás mellett szerepeltetem és alapesetben az ideiglenes vásárló lehetőség van kiválasztva. Ezen eleme a HTML-nek hasonló módon működik, mint a már megismert szöveg mező. A hivatkozási név használatával a következő oldalon a megfelelő PHP kóddal beolvasom a választás eredményét és felhasználom az oldal forráskódjában a megfelelő helyen. A radio gomb minden lehetősége külön értékkel rendelkezik az azonosíthatóság végett. Ezt követően bekérem az előző lehetőséget figyelembe véve a személy tagsági számát vagy az ideiglenes azonosítóját. Ezt egy egyszerű szöveg mezővel oldottam meg. A tagsági státusz és az azonosító páros egyértelműen meghatározza a vásárló személyazonosságát a rendszerben. Ezen megoldással gyorsabban lehet egy kölcsönzést elindítani, mint a személyes adatok megadásával. Ha az illető nem ismeri az azonosítót a keresés menüpontban könnyen visszakereshető név és születési idő alapján. A tagsági szám szerepelhet egy esetleges tagsági kártyán, a ténylegesen létrehozott kölcsönző esetében.

A következő kitöltendő mező egy lista, amely a telephelyen található kerékpárok sorszámait tartalmazza, melyből kijelöléssel választhatjuk ki a kívánt elemeket. Ezen lista szintén HTML elem, de az adatok, melyekkel feltöltöm, azok PHP kódból származnak. A lista a SELECT elem használatával készül, melynek visszatérési értéke egy tömb lesz, mivel nem egy elem kiválasztására van korlátozva a kijelölés, hanem engedélyezett a korlátlan számú elem kijelölése. A lista elemei dinamikusan változnak, az adott helyzetnek megfelelően. A lista PHP utasításokkal készül, mivel az elemeket a kerékpárok tábla állapot oszlopának lekérdezésével kapom meg. A bejelentkezett eladóhoz hozzárendelt telephely számát a megfelelő session változóból nyerem ki, melyet a lekérdezés során felhasználok. A következő kód egy PHP blokkon belül található, de észrevehetően tartalmaz HTML utasításokat. Ezeket

a PHP nem tudja értelmezni, tehát az oldal forrásába kell megjelenniük, hogy a célszámítógép böngészője fel tudja használni az oldal létrehozásához őket. Ezt úgy érhetjük el, hogy a print utasítást használva a kimenetre iratom a HTML elemeket a PHP blokkon belül, melyeket így a böngésző értelmezni tud majd. A kód első sorában lekérdezem azoknak a kerékpároknak a sorszámát, melyek a $telep változóval azonos telephelyen találhatóak. Ezek azok a kerékpárok, melyek jelenleg az eladóval azonos telephelyen vannak és így kiadhatóak kölcsönzésre. Ezután a print és egy while ciklus segítségével létrehozom a listát a lekérdezés eredménytáblájának segítségével. Megfigyelhető a string összefűzés, melynek segítségével a kimentre kerül azon HTML kód, melyet értelmezni tud a böngésző. A HTML kifejezések idézőjelek között szerepelnek, amíg a PHP változók azok nélkül, melynek eredményeképp a kimenetre a változó értéke kerül és nem a változó neve. A stringeket a pont karakter fűzi össze. A SELECT elem értékeit az OPTION kezdő és záró jele között kell szerepeltetni. Az értékét a sornak az aktuális sorszám adja, melyet listaelemként szerepeltetünk is. A kód a következő:

```
$eredmeny=mysql_query("SELECT sorszam FROM kerekpar WHERE allapot='$telep'");
print "<SELECT NAME=\"sorszam[]\" MULTIPLE SIZE=5>\n";
while($egy_sor=mysql_fetch_array($eredmeny)){
print "<option value=".$egy_sor['sorszam'].">".$egy_sor['sorszam']."\n";
}
print "</select>";
```

A kölcsönzést a Kiad gomb megnyomásával indíthatjuk el. Ekkor a feldolgozó oldal forráskódjában átvesszük a HTML űrlap változóinak az értékeit. Ha nincs megadva tag/ideiglenes azonosító vagy nincs kiválasztva kerékpár sorszám a listából, akkor kimenetre egy hibaüzenet kerül és a forráskód futása leáll, azaz nem hajtja végre az adatbázison a kijelölt műveleteket hiányos adatokkal. Ezzel az ellenőrzéssel az adatbázis egységessége és a nyilvántartó rendszer működésének hatékonysága is megőrizhető. Ha az oldal megkapta a szükséges változókat, akkor két adatbázis műveletet kell végrehajtani, hogy a kölcsönzés létrejöjjön. Az első műveletben létrehozok egy új sort az adatbázis kölcsön táblájában a kapott adatokkal.

A bevitt adatok a következőek:

- A személy tagsági státusza
- Az azonosító száma
- Kerékpárok sorszáma
- Aktuális telephely

Ezen adatokon kívül a sorba kerül még az aktuális idő és egy állapot változó, mely a kölcsönzés lezárásához szükséges. A több kerékpár kölcsönzése esetén előálló több sorszámot a táblában egy string összefűzéssel kezelem, melynek során a sorszámokat a „-” karakter segítségével összefűzöm és ezt a karaktert, mint elválasztót használom a sorszámok között. Ezen megoldással az adatbázis nem teljesíti a harmadik normálforma követelményeit, de átláthatóbb adminisztrációs szempontból. Az aktuális időt egy időbélyeg használatával oldom meg, de ezt bővebben az adatbázis bemutatása fejezetben fejtem ki.

A második adatbázis művelet, amit el kell végezni a kölcsönzés létrejöttéhez, a kiválasztott kerékpárok helyzetének reprezentálására szolgáló állapot változójának megváltoztatása. A jelenlegi implementációban a „0” érték jelöli a kikölcsönzött állapotot, amíg az [1-5] intervallum a jelenleg létrehozott telephelyek azonosítószámait jelölik. Ezen azonosítók lekérdezésével és az aktuális telephely azonosító számának egyeztetésével írattam ki a kikölcsönözhető kerékpárokat. Ahhoz, hogy a következő kölcsönzés indításánál ne jelenjenek meg ezen kerékpárok, az állapotot 0-ra kell állítani, de csakis a kiválasztott sorszámú kerékpárok esetében. Ez a következő kóddal történik:

```
for ($i = 0; $i < $meret; $i++) {
   $result=mysql_query("UPDATE kerekpar SET allapot='0' WHERE
sorszam='$sorszam[$i]'");
}
```

Az UPDATE paranccsal nem hozunk létre új sort, mindössze a létező sorokban ahol a sorszám megegyezik az általunk kiválasztott sorszámmal, az állapotot 0-ra állítjuk be. A $meret változó a sorszám tömb méretét reprezentálja, a for ciklus 0-tól indul, mivel a tömb első elemét a 0 indexszel érhetjük el. Ezen ciklus lefutása után a kölcsönzés létrejöttnek tekinthetjük minden szempontból. Ha a kód nem érzékel hibát a lefutás során, akkor egy sikerült üzenetet jelenítünk meg a kimeneten, így tudatva az eladót, hogy sikeresen regisztrálásra került az általa elindított kölcsönzés, tehát fizikailag is kiadhatja a kiválasztott sorszámú kerékpárokat a vásárlónak.

4.3.3 Kölcsönzés lezárása

A kölcsönzés lezárása bármely telephelyen lehetséges, nem szükséges azon a telephelyen megtenni, amelyen a vásárló kikölcsönözte a kerékpárt. Az eladó a visszavétel menüpontot

kiválasztva kezdheti meg a folyamatot. A kerékpárt nem kötelező a kikölcsönző személynek leadni, ezt megteheti egy megbízott személy is, mivel a kölcsönzés lezárásakor nem tároljuk le a leadó személy adatait.

A munkafelület itt két fő részre bomlik. A felső részen megjelennek kilistázva az aktív, azaz még le nem zárt kölcsönzések. Ezen kölcsönzések kiíratása a már bemutatott módon történik, melynek folyamán az adatbázisból történő adatok kiolvasása után a PHP blokkon belül HTML utasításokat hozunk létre és a kapott adatokkal töltjük fel azokat. A kölcsönzések adatai egy táblázatba rendezve helyezkednek el, melynek oszlopai a kölcsönzés azonosítóját, a kölcsönző személy tagsági státuszát és azonosító számát, a kerékpárok sorszámát és a kölcsönzés megkezdésének idejét tartalmazzák.

A fennmaradó részen egy HTML űrlap helyezkedik el, melynek segítségével kiválaszthatjuk egy listából, mely azonosítóval rendelkező kölcsönzést szeretnénk lezárni. Ezt a listát szintén egy adatbázis lekérdezés eredményéből hozom létre, ezzel biztosítva, hogy csakis lezáratlan kölcsönzéseket lehessen kiválasztani ebben az opcióban. Lényeges kérdés merül fel abban az esetben, mikor több kerékpárt kölcsönöznek ki, de csak egy részüket kívánják leadni az adott időben és a fennmaradó kerékpárok kölcsönzését továbbra is folytatni kívánják. Erre az eshetőségre egy listából kiválasztható két opciót hoztam létre, melyek a mindegyik és az egy részük nevet viseli. Egyértelműen a mindegyik opciót kiválasztva minden a kölcsönzésben szereplő kerékpárt visszavételezünk és lezárjuk a kölcsönzést. Az egy részük opciót kiválasztva megjelenik, egy újabb lista melyből a kiválaszthatjuk, mely kerékpárok kölcsönzését kívánja folytatni a vásárló.

A két opció két PHP oldalra irányítja át az eladót, melyek szerepe az adatbázis műveletek végrehajtása és a kölcsönzési adatok kiíratása, köztük a kölcsönzés idejének megállapítása. A mindegyik opciót kiválasztva a PHP oldal forrásában az első teendő az űrlapon található változók kiolvasása. Ezután a megfelelő kölcsönzést kiválasztva az adatbázisból az adatokat változókba tárolom. A sorszámok beolvasása a „-” határoló elem segítségével egy tömbbe valósul meg, melynek folyamán a határoló jelig beolvasom a stringet, aztán a kapott értéket letárolom és folytatom a string beolvasását. Az adatok kinyerése után a kitöltetlen mezők értékeit az UPDATE segítségével átállítom a megfelelő értékekre. Ezek a mezők a kölcsönzés befejezésének az időpontja és a leadás helye, ezen kívül az állapot mező értéket nulláról egyre változtatom, így lezártnak tekintve a kölcsönzést. A kölcsönzés idejének maghatározásához egy saját függvényt készítettem, mely kezdeti paraméterként a két időpontot kéri be, és

kimenetként egy tömböt hozz létre, mely négy cellával rendelkezik, melyek a napot, órát, percet és másodpercet tárolják. Mivel az adatbázisba történő íráskor, nem a PHP kód által adom meg a kölcsönzés idejének végét, hanem egy időbélyeget rendel hozzá a MySQL az adott mezőhöz, ezért egy lekérdezés segítségével olvasom ki az időpontot és így tudom azt paraméterül adni a függvénynek. További teendő a kerékpárok helyzetét jelző állapot mező megfelelő értékre történő beállítása. Ezt az értéket az aktuális telephely azonosítójának értékére állítom be, a megfelelő sorszámú kerékpároknál. Ezzel a lépéssel biztosítom, hogy a kerékpárok mostantól a leadás helyén szerepelnek az adatbázisban és kölcsönözhetőként jelenjenek meg a kiadás menüpontban. A kölcsön tábla lezárása után lekérdezem a kölcsönző személy adatait és azokat a kölcsönzött kerékpárok számával, a kölcsönzés megkezdésének idejével és a kölcsönzés tényleges idejével, melyet nap, óra és perc formátumba iratok ki, valamint egy sikerült üzenet kíséretében iratok ki. Ezen információk alapján az eladó könnyedén megszabhatja a kölcsönzés díját és visszavételezheti a kerékpárokat.

Az egy részük opciót kiválasztva és a következő űrlapon megadva a listából, hogy mely kerékpárokat nem hozta vissza a jelenlegi kölcsönzésből a vásárló, a PHP kód egy újabb oldal betöltését végzi el. Ezen oldal funkciója, nagyban hasonlít az előzőekben bemutatottra, de a jelen helyzetben nem lehetséges minden kerékpár visszavétele a rendszerbe. Mivel az adatbázist úgy alakítottam ki, hogy egy kölcsönzéshez egy sor tartozik, nem lehetséges szétválasztani a kölcsönzött kerékpárokat az adott kölcsönzésazonosító alatt. Az itt található algoritmus első fele megegyezik a mind opciónál bemutatottal, de a kölcsönzés lezárása után több műveletet is elvégez. Az adatok kinyerése és a kölcsönzés lezárása után, létre kell hozni egy új kölcsönzést, melybe tárolom a fennmaradó kerékpárokat, így biztosítva a kölcsönzés folytatását. Ezt az új kölcsönzést a lezárt kölcsönzés adataival indítom el, de a fennmaradó kerékpárok sorszámával. Így létrejön egy kölcsönzés, melynek során a kölcsönző személye, a kiadás telephelye és a kezdés ideje nem változik, de a kölcsönzött kerékpárok azok lesznek, melyeket a második űrlapon megadott az eladó. További különbség, hogy csak a visszavételezett kerékpárok állapotazonosítóját állítom a jelenlegi telephelyre, azokat melyek kölcsönzését folytatjuk, azokét hagyom a nullás állapotban. Ez az opció is a megfelelő adatok kiíratásával végződik, melyeknél figyelembe kellett vennem a kerékpárok számának pontos meghatározását.

4.3.4 Keresés az adatbázisban

Az utolsó bemutatásra kerülő menüpont az áttekintés, melyet eredetileg egy egyszerű telephely állapot áttekintésre terveztem, de a kódolás során felmerült az igény több keresési megoldásra, melyeket szintén ebbe a menüpontba implementáltam. Mivel az eladók nem fognak rendelkezni azon joggal és lehetőséggel, hogy közvetlenül az adatbázisba tekintsenek be, ezért többféle keresési opciót is készítettem, hogy a felmerülő igényeket a legnagyobb mértékben tudja a program kielégíteni egy esetleges valós kerékpárkölcsönző megvalósítása esetén. Ebben a fejezetben bemutatom a kereséseket, melyek négy fő csoportot alkotnak.

Az áttekintés menüpont elrendezése megegyezik a regisztráció menüpontéval. A munkaterület fölött helyezkednek el a fő csoportok linkjei, melyek segítségével választhatjuk ki, milyen keresést szeretnénk végrehajtani. Az első opció a telephely aktuális állapota, mely az első elképzelésem szerint a teljes menüpont funkcióit lefedte volna. Ezt az opciót kiválasztva egy információs oldalt kapunk, melyen nem adhatunk meg semmilyen adatot, tehát nem vagyunk befolyással a megjelenő adatokra és a lezajló folyamatokra. Ezen oldal funkciója az, hogy az eladó ellenőrizni tudja, hogy a telephelyen lévő állapot és az adatbázis tartalma azonos-e. Ezen oldalon a kiválasztását követően megjelenik a telephelyen található kerékpárok száma, azok sorszáma és a kikölcsönzött kerékpárok sorszáma. Ezen adatokat a kerékpárok tábla állapot oszlopának vizsgálatával érem el. Egy egyszerű lekérdezés után az adatokat változókban tárolom és a megfelelő HTML elemekkel rendezve kiíratom táblázatok segítségével, a könnyebb olvashatóság kedvéért. Ha az oldal nem tudja elérni az adatbázist, akkor hibaüzenet jelenik meg, mely erről tájékoztatja az eladót, aki értesíteni tudja az oldalt felügyelő személyzetet, akik kijavíthatják a hibát.

A következő opció a foglalások nevet viseli és azon problémára megoldás, hogy a felhasználók által regisztrált online foglalásokról, hogyan értesülnek a telephelyen dolgozó személyek. A leadott foglalások a foglalás táblában tárolódnak el, de az adatbázishoz nem férnek hozzá az eladók, így kellett egy hely ahol megtekinthetik a foglalásokat. A keresések között megfelelő helyre helyeztem el őket, mivel itt két kattintást követően megtekinthetik milyen foglalásokat kell teljesíteniük és mikor. A foglalások ki listázásánál arra is ügyelnem kellett, hogy a már lejárt, teljesített foglalásokat nem célszerű továbbra is megjeleníteni, mivel egy idő után rengeteg helyet fognak elfoglalni a képernyőn ezek a továbbiakban értéktelen adatok. Ezért a listázás után bekerült egy HTML űrlap melybe megadhatják az eladók, mely foglalást rögzítették, azaz arra az időpontra a megfelelő számú kerékpárt biztosították. A

regisztrálás gombra kattintás után a következő oldal forrásában található kód csatlakozik az adatbázishoz és a foglalás tábla állapot oszlopában a megfelelő sorban az értéket 1-re állítja át. Ezen változtatás után a kiválasztott foglalás nem fog megjelenni a listázásnál, így felszabadítva a helyet a további foglalások előtt.

A kölcsönzések lezárása után nem volt lehetőség, hogy az eladók megtekintsék azokat, így a harmadik keresési lehetőség a kölcsönzések közötti keresés lett. A kölcsönzések adati menüpont alatt található funkció egy HTML űrlappal kezdődik, amely az aktuális munkafelületen helyezkedik el. Ennek a funkciónak az érdekessége, hogy az űrlap nem adja át a vezérlést egy következő PHP oldalnak, hanem önmagára mutat. Az ezt szolgáló kódrészlet a következő: <form action=<?php print $_SERVER[PHP_SELF]?> method="POST">. A forráskódban megtalálhatóak az űrlap változóinak átvétele és a velük történő munka is. Az első meghívása után az oldalnak ezen változói nem kapnak értéket, mivel még nem történt meg az űrlap kitöltése. A további adatbázis művelteket egy olyan feltételez kötöttem, hogy a vizsgált változók rendelkeznek-e értékkel, ha igen akkor lefutnak a hozzá tartozó műveletek, ha nem akkor nem történik változás. Az űrlap négy kitöltendő területtel rendelkezik, amelyek külön-külön is funkcionálnak, de tetszés szerint is kitölthetők bármely párosításban. A keresési lehetőségek a következők:

- Dátum szerinti keresés
- Időintervallum szerinti keresés
- Kerékpár szerinti keresés
- Azonosító szerinti keresés

A dátum szerinti keresés esetében az adott napon regisztrált kereséseket listázza ki az oldal. Az időintervallum esetében egy tól-ig határ között szereplő kölcsönzések jelenek meg. A kerékpár szerinti kölcsönzés esetén a megadott sorszámú kerékpárhoz tartozó összes keresést megjeleníti az oldal. Az azonosító szerinti keresésnél pedig a megadott azonosítóval rendelkező kölcsönzést fogjuk látni a képernyőn.

Az utolsó elérhető keresési opcióban a felhasználók keresése történik. Ezen opció akkor válik fontossá, ha egy vásárló már használta a szolgáltatást, azaz már szerepel a rendszerben, de nem emlékszik a tagsági számára vagy az ideiglenes azonosítóra. Ezen információk ismerete nem is várhatók el a vásárlóktól, így egy egyszerű név és születési idő megadásával az eladó megkeresheti a személyhez tartozó adatokat és azokkal elindíthatja a kölcsönzést. Ennek az oldalnak a működése is egy HTML űrlappal kezdődik, melyen megadhatjuk a keresésünknek megfelelő adatokat. Háromféle keresés indítható, melyeknél a személy nevét és születési

idejét, a tagsági számát vagy az ideiglenes azonosítóját tudjuk és keressük a hozzájuk tartozó további adatokat.

Ezzel a keresési funkcióval kiegészítve az áttekintés menüpont egy jól strukturált, könnyen átlátható és kezelhető keresési felületet biztosít az eladók számára, melynek segítségével adatokat kérhetnek le a számukra szükséges módon az adatbázisból, amelybe ezen kívül a funkción kívül nincs jogosultságuk betekinteni.

5. Összefoglalás

A bemutatott elméleti anyag segítségével nincs szükség nagymértékű programozói tudásra, ahhoz hogy a szakdolgozatban szereplő programkódokat értelmezni lehessen, így értelmezhető olyan emberek számára is, akik nem jártasak a PHP vagy más szkriptnyelv szintaktikájában.

A kerékpárkölcsönző bemutatása során mindössze egy lehetséges gyakorlati megvalósítását mutatattam be a kölcsönzőnek, melyet a program elkészítésekor használtam fel. Egy tényleges megvalósításkor a mindenkori igényeknek és erőforrásoknak megfelelő testre szabása végezhető el a kerékpárkölcsönzőnek, mely így a legjobban felelhet meg a felé irányuló piaci igényeknek.

A program implementációjában sikeresen megvalósítottam a bevezetésben és a cég a bemutatásával foglalkozó fejezetben felvetett igényeket. Megvalósítottam egy olyan nyilvántartórendszert, mely képes tetszőleges számú telephelyet kiszolgálni azonos időben, mégis elkülönítve egymástól őket. Közös kerékpárállománnyal és közös vásárlóbázissal rendelkeznek, mégis nyomon lehet követni mely telephelyről érkezett az adott adatbázis lekérés vagy esetleges módosítás. A beépített login rendszer segítségével és a további titkosítások használatával a rendszer képes azonosítani az azt felhasználó személyeket és biztosítani a bizalmas adatok védelmét. További fejlesztésre ad lehetőséget a program minimális grafikus felülete, melyet a különböző webes szabványokkal könnyen lehet cserélni, ezzel alkalmazkodva a különböző megvalósítások különböző igényeihez. A program bizonyos részeiben lehetőség van további hibakezelések beépítésére, melyek növelhetik a stabilitását és felhasználhatóságát a rendszernek. Nagymértékű fejlesztésre adhat okot egy számlázási opció beépítése, mellyel nagyban megnövelhető lenne a program felhasználásának a köre. Ezzel és

további funkciók beépítésével elérhető lenne, hogy a program teljesen lefedje a kölcsönző működése közben felmerülő feladatokat.

A nyilvántartó rendszer a jelenlegi implementációjában képes a kölcsönző működéséhez szükséges nyilvántartási feladatokat ellátni, bármely web szerverre történő telepítés esetén, az adatbázis kapcsolat megteremtését követően.

A kerékpárkölcsönző gyakorlati megvalósításának gazdasági és informatikai szempontok szerinti részletes kidolgozása és a program további bővítése kitűnő lehetőséget nyújt egy általam megszerezni kívánt Msc diploma szakdolgozatának témájához.

A nyilvántartórendszer és a bemutató weboldal elérhetőek az alábbi címeken:

- http://mecsopc.inf.unideb.hu/~komlosid/nyilvantarto/
- http://mecsopc.inf.unideb.hu/~komlosid/bemutato

A nyilvántartó rendszerbe való belépéshez, valamint az online foglalás eléréséhez a következő felhasználói név, jelszó páros használható: admin-admin.

6. Irodalomjegyzék

George Schlossnagle: PHP fejlesztés felsőfokon

Joe Celko: SQL felsőfokon

Julie C. Meloni: A PHP, a MySQL és az Apache használata

Julie C. Meloni: Tanuljuk meg a MySQL használatát 24 óra alatt

Matt Zandstra: Tanuljuk meg a PHP5 használatát 24 óra alatt

Peter Moulding: PHP Haladóknak Fekete Könyv

Sági Gábor: Webes adatbázis-kezelés MySQL és PHP használatával

http://www.dev.mysql.com/

http://www.developertutorials.com/category/tutorials/php/

http://www.java2s.com/Code/Php/CatalogPhp.htm

http://www.php.net/

http://www.phpeasystep.com/

http://www.php-script.hu/index.php

http://www.phpsimple.net/

http://www.tizag.com/

http://www.tuxradar.com/practicalphp

http://www.w3schools.com

http://www.webdesign.org/web-programming/php/page-1.html

http://www.weblabor.hu/

http://www.webstockbox.com/php/

http://www.wikipedia.org/