Diplomamunka

Nagy Zsolt

Debrecen

2009

Debreceni Egyetem

Informatika Kar

Adatbázis oktatást segítő Web alapú oktatási anyag fejlesztése

Témavezető

Dr. Adamkó Attila

egyetemi tanársegéd

Készítette:

Nagy Zsolt

informatika tanár

Debrecen

2009

1. Tartalomjegyzék

2. Ábrajegyzék

3. Mellékletek jegyzéke

4. Bevezetés

A célom egy adatbázis tanítást/tanulást segítő oktatási anyag elkészítése, az informatika szakmacsoportos képzésben résztvevő középiskolások számára. A tanítás során úgy tapasztaltam, hogy ennek az anyagnak az elsajátítása nehéz, és nagyon szélsőséges abban az értelemben, hogy egyesek ráéreznek és képesek kielégítő módon elsajátítani, mások pedig nagyon hamar elvesztik a fonalat és a továbbiakban csak küszködnek. Rajtuk szerettem volna segíteni akkor, amikor egy diasorozatot készítettem, ábrákkal és kiegészítő magyarázatokkal. A hagyományos „krétás" táblán való prezentáció lassabb és kevésbé látványos mint egy előre megtervezett és elkészített diakép. Ezért az elméleti órákat ilyen elektronikus formában sikerült megtartani. Természetesen az elméletet követő feladatokat már a táblánál oldottuk meg. Ezek után vetődött fel bennem azaz elképzelés, hogy készítsek egy webes felületű segédanyagot, először az adatbázis-kezeléshez, majd a jövőben a programozás és a hardver tanításához. A diplomamunka során ennek az oktatási anyagnak az elkészítését szeretném ismertetni.

Az elkészített Webhely címe: http://tanulnijo.uw.hu

5. A fejlesztés szempontjai

Amikor szoftverfejlesztésre adjuk a fejünket, követnünk kell bizonyos leírásokat. A szoftverkrízis korszakának nemcsak az volt a hibája, hogy mennyiségileg nem tudták kielégíteni az igényeket, hanem az is, hogy az elvárt minőséget sem érték el. A mennyiségen segítettek a 3. generációs programozási nyelvek, programkönyvtárak, majd pedig a 4. generációs programfejlesztő eszközök létrehozásával. Ilyen korszerű nyelv a C++, Java, Object Pascal és az ezekre épülő vizuális eszközök, mint a Visual Studio, Delphi. Ezek az eszközök űrlap- és jelentéstervezővel, adatbázis kapcsolási felülettel, hibakereséssel gyorsították a fejlesztést. A minőséget különböző program-, szoftver- és rendszerfejlesztési módszerekkel és módszertanokkal támogatták. Nagy előrelépés volt a strukturált programozás bevezetése. A program kódot részekre bontották, tagolták: vertikálisan alprogramokkal, horizontálisan modulokkal. Így hordozható programozási elemeket, könyvtárakat tudtak készíteni. Egy általánosan megírt alprogramot máshol is fel lehet használni, és egy jól letesztelt, hibátlan kóddal biztonságosabb termék készíthető. Kifejlesztették az objektum-orientált programozást, amellyel többek között a programkészítés folyamata egy fokkal közelebb került ahhoz a szemlélethez, ahogy az ember a környezetét (dolgokat, objektumokat, viselkedéseiket) érzékeli. Főleg közepes vagy nagy szoftvertermékek készítésénél fontos, hogy valamilyen fejlesztési módszertan leírásait kövessük. Ilyen az SSADM (Strukturált Rendszer Elemzési és Tervezési Módszertan), illetve az UML (Egységes Modellező Nyelv). Nagyobb projekteknél az informatikai fejlesztés csak egy részét képviseli a megoldandó problémának, ezért lehetővé kell tenni az informatikai fejlesztés integrációját a teljes folyamatba. Ilyen befogadó módszertan a PRINCE projektirányítási módszertan.

Amikor le akarunk írni valamilyen bonyolultabb dolgot, folyamatot, akkor a modellekhez nyúlunk. Ezek segítenek abban, hogy a valós világ bonyolultságát kezelhető méretűre tudjuk zsugorítani. Ehhez ki kell szűrnünk a valóság azon elemeit, amelyek a megvalósítandó rendszerünk szempontjából irrelevánsak. Ilyen modelleket használnak például a szoftverek adatszerkezetének leírásához (adatmodellek), és a funkciók, viselkedések leírásához (folyamat és esemény modellek).

Minden szoftver egyszer megszületik, majd használják valameddig és végül lecserélik, hogy a helyét átvegye egy fejlettebb, az akkori követelményeket pontosabban kielégítő új termék. Amikor elkészítünk egy szoftvert, akkor mindig az akkori környezet igényeit

próbáljuk teljesíteni. Viszont a világ változik és így mindig újabb és újabb elvárásoknak kell megfelelni. Vagyis a programok is ciklikusan változnak, ezért alakultak ki az úgynevezett életciklus modellek (vízesés életciklus modell, V-modell, és Boehm spirál életciklus modellje).

Életciklus modell részei:

- Elemzés, követelmény meghatározás
- Rendszer tervezés, részletes tervezés
- Kódolás
- Modul tesztelés
- Részletes tesztelés
- Dokumentálás

Az egyszerűbb szoftvereknél nem kell szétválasztani a teljes rendszer és a részrendszer tervezését és modulok híján a modul tesztelés is elmaradhat.

6. Elemzés, követelmény meghatározás

Ennek a szakasznak a feladata, hogy feltárjuk a jelenlegi rendszer működési körülményeit, szerkezetét, folyamatait. Meghatározzuk, hogy mit kéne változtatni ahhoz, hogy egy jobban működő, az igényeket jobban kielégítő alkalmazást kapjunk.

6.1. A jelenlegi helyzet

Az adatbázis tanításhoz egy diasorozatot használok, amely jellegéből adódóan vázlatos képet szolgáltat. Ez megfelelő az órai feldolgozás során. Viszont az otthoni felkészülést jobban támogatná egy olyan média, amely hosszabb magyarázatokat is megenged. Fontosnak tartom még a szövegen belüli hivatkozási pontok létrehozását, vagyis a hipertext jellegű dokumentumkezelést. Illetve nem utolsó sorban legyen mindenki számára könnyen hozzáférhető. Ezen okok miatt döntöttem a webes felület mellett.

6.2. Célmeghatározás

Az oktatási anyag három területet érint. A legnagyobb rész az elmélet ismertetésére vonatkozna, képekkel, magyarázatokkal ellátva. A következő rész az elméleti anyag megértését, rögzítését szolgáló feladatokból, gyakorlatokból állna. Majd a tudásunkat egy ellenőrző feladatsorral tesztelhetnénk.

7. Tervezés

7.1. A tartalom leírása

A webhely elsődlegesen három részből áll:

- adatbázis-kezelés,
- programozás és
- hardver.

A szakdolgozat során az adatbáziskezeléssel foglalkozó részt szeretném megvalósítani, amely a következők szerint tagozódik:

- elmélet,
- gyakorlat,
- teszt.

Az elmélet az adatbáziskezelés alábbi fejezeteivel foglalkozna:

- adat, információ,
- adatbázis, adatbáziskezelés, adatbáziskezelő-rendszer, adatbázisrendszer,
- rendszer, modell, adatmodell,
- egyed-kapcsolat modellezés,
- relációs adatmodell,
- egyed-kapcsolat modell átalakítása relációs modellé, normalizálás,
- relációs algebra,
- Access adatbáziskezelő-rendszer ismertetése,
- SQL nyelv ismertetése

A gyakorlati részben, olyan feladatokat, gyakorlatokat tervezek, amelyek segítenek megérteni és elmélyíteni az elméleti anyagot.

A teszt kérdései az elméletre épülnek. Választhatunk, hogy az egyes fejezetekből vagy a teljes anyagból vegyesen kerüljenek összeállításra.

7.2. A kezelő felület leírása

A weboldalak felépítése mind a három résznél hasonló lesz:

- fejléc: Ebben helyezem el a webhely logóját, a fő- és almenüket.
- törzs: A legnagyobb területet foglalja el a tartalom.
- lábléc: Itt megismétlem a könnyebb kezelhetőség miatt az almenüt, illetve kiegészítő információkat helyezek el (e-mail cím, utolsó módosítás dátuma).

1. ábra: Az oldal szerkezete

Az elméletnél és a gyakorlatnál a tartalomjegyzék ki lesz emelve egy-egy oldalra a könnyebb kezelhetőség miatt.

A fő navigációs sáv felépítése:

- home: A kezdő oldalra visz.
- adatbázis: Az adatbáziskezelés elméletével foglalkozó rész tartalomjegyzékére visz.
- programozás: Jövőbeni fejlesztés.
- hardver: Jövőbeni fejlesztés.

Az alnavigációs sávok a webhely egyes részeinek navigációit fogják ellátni, de a felépítésük egységes, szintén a könnyebb kezelhetőség miatt:

- elmélet: Az egyes témakörök elméleti fejezeteinek tartalomjegyzékéhez visz.
- gyakorlat: Szintén témakörön belül, de a gyakorlati fejezetek tartalomjegyzékéhez jutunk a segítségével.
- teszt: Témakörön belül a tesztoldalra mutat.
- tartalomjegyzék:
- előre, következő: A fejezetekben mozoghatunk előre vagy vissza.

A kisegítő sáv felépítése:

- elérhetőség: A készítőnek szóló e-mail küldési lehetőséget biztosít.
- utolsó módosítás dátuma: Mikor lett utoljára frissítve a webhely.

7.3. A teszt viselkedésének leírása

Ezen a weblapon dinamikusan előállított feladatlapot kapunk. Ehhez előbb választani kell a kategóriák közül. A feladatok kategóriákba sorolása az elmélet fejezeteinek megfelelően történik. Lehetőség van a teljes elméleti anyagból, vegyes feladatsor összeállítására. Az utóbbiból tíz, az előbbiből öt kérdésből áll egy feladatsor. Az ellenőrzés után kapunk egy értékelést, amely tartalmazza az elért és az összes pontot, az eredményt százalékosan, illetve a feladatok helyes megoldásait. Vagyis lesz lehetőségünk tételesen ellenőrizni a megoldásaink

helyességét. Az eredményeket egy külön kisablakban kapjuk, így könnyebb az ellenőrzést elvégezni.

7.4. A teszt hátterét adó adatmodell megtervezése

Nyilván akarom tartani a feladatokat és a válaszokat. Mivel a feladatokhoz változó számú válasz tartozik, ezért külön egyedként veszem fel őket.

A „feladatok” egyedet jellemző tulajdonságok:

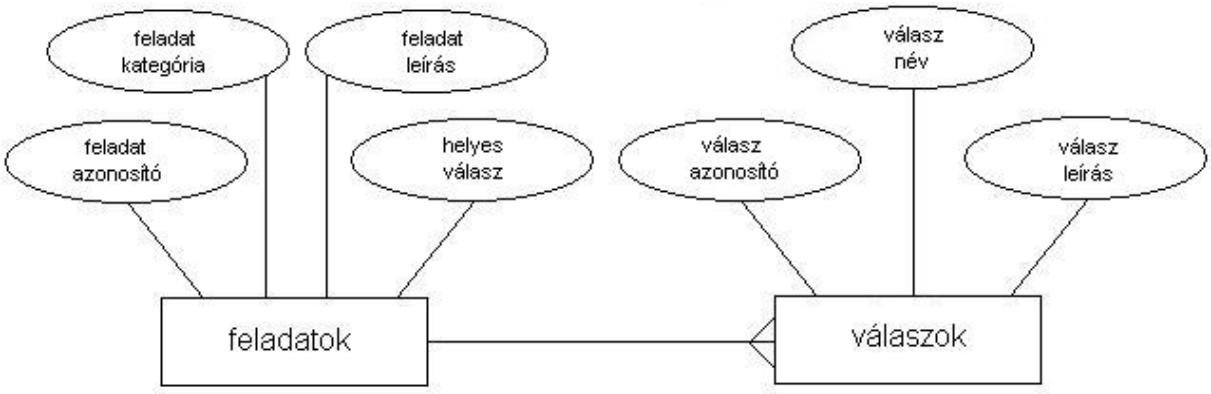
- feladat azonosító: A konkrét feladat azonosítója.
- feladat kategória: Az elméletnek megfelelően melyik fejezethez tartozik.
- feladat leírás: A feladat megfogalmazása, vagyis egy konkrét kérdés.
- helyes válasz: A válaszok közül melyik a megfelelő {’a’, ’b’, ’c’, ’d’}.

A „válaszok” egyed jellemző tulajdonságai:

- válasz azonosító: Az adott válasz azonosítója.
- válasz név: Az {’a’, ’b’, ’c’, ’d’} halmazból kikerülő érték.
- válasz leírás: Az adott feladatra megfogalmazott egyik válasz leírása.

Mivel egy feladatok egyed-előforduláshoz több válaszok egyed-előfordulás is tartozhat, és egy válaszok egyed-előfordulás csak egy feladatok egyed-előforduláshoz, ezért kapcsolat fajtája egy:több típusú a feladat:válasz egyed-előfordulások között. A kapcsolat bináris, mivel két egyedet köt össze. További jellemzője, hogy mind a két irányból teljes, vagyis mindegyik egyed egyed-előfordulása kapcsolódik a másik egyed valamelyik előfordulásához.

Ezért az egyed-kapcsolat modell a következőképpen néz ki:



2. ábra: A feladatok – válaszok egyedek EK modellje.

8. A felhasznált technológiák rövid leírása

8.1. Internetes alkalmazásfejlesztés

Kezdetben a Web segítségével csak szöveges dokumentumok, statikus HTML felületen való szolgáltatása volt a cél. Mára az Internet egy megkerülhetetlen kommunikációs közeg, az információgyűjtés, és a szórakozás tere egyaránt. Nem nélkülözheti a grafikus elemeket, a különböző médiát. Az internetet használók képeket, videókat akarnak nézni és zenéket hallgatni. Lényegesen kibővült a célközönség; a kutatásait megosztani akaró fizikusok mellett jelen vannak a hirdetéseiket megjelentető cégek, az emberek közzé tehetik blogok formájában a gondolataikat, a távol élő nagyszülő pedig megnézheti az unoka szülinapi videóját. Mindez csak úgy valósulhat meg, ha a Web technológiai háttere követi az igényeket. Persze az is lehet, hogy a Web újabb lehetőségei alakítanak ki az emberekben szükségleteket. Ma gyakran lehet hallani azt a kifejezést, hogy Web 2.0. A szám nem verzió számot, hanem második generációt jelöl, az Internet evolúciójának aktuális lépcsőfokát. Régóta használt szabványos technológiák összekapcsolásával, olyan minőségű alkalmazásokat tesz elérhetővé a Web-en, amilyet eddig csak az operációs rendszer desktop-ján futó alkalmazások biztosítottak, és lehetőséget teremt mindenkinek arra, hogy tartalom szolgáltató legyen. A résztvevők közösen alkothatják, és megoszthatják a tartalmakat, miközben valódi interaktív kommunikáció zajlik.

8.1.1. A Web architektúra felépítése

Egy internetes alkalmazás három elkülönülő szolgáltatásra bomlik. Az első szelet egy prezentációs réteg, amit a háttérben egy internetes böngészőprogram támogat. Az ebben a szolgáltatásban szereplő funkciók a felhasználó számítógépén futnak. Ilyen böngészők a Mozilla Firefox, Google Chrome, Opera, Safari, Netscape Navigator, Internet Explorer. Az első mérföldkő az NCSA Mosaic for X 0.10-es 1993-ban kiadott grafikus felületű böngészője volt, amely elérhetővé tette a hétköznapi ember számára is a Web-et.

A második szelet az üzleti logikát megvalósító programrész. Az itt szereplő komponensek (PHP scriptek, JSP alkalmazások) a Web-szerveren futnak. A Web-szerver, mint amilyen az ingyenes Apache, egy internetes szolgáltatást biztosító szoftver.

A Web architektúra harmadik szeletét az adatbázis-kezelés alkotja. Ezeket a feladatokat általában egy külön szerver, az adatbázisszerver, például a szintén ingyenes MySQL valósítja meg.

A szolgáltatás-alapú architektúra működési folyamata alapvetően a felhasználói kérések hatására indul. Ilyen kérés lehet egy hiperhivatkozásra való kattintás, vagy egy nyomógomb lenyomása. Ennek hatására egy HTTP-kérés indul el a Web-szerver felé. A Web-szerver pedig értelmezi a kérést, és végrehajtja a szükséges műveleteket. Ami általában egy válasz üzenet formájában ér véget, és a kliensoldali böngészőprogram jeleníti meg.

Attól függően, hogy a fenti szolgáltatást megvalósító alkalmazások ugyanazon, vagy esetleg több gépen helyezkednek el, beszélhetünk egyrétegű, illetve többrétegű alkalmazási modellekről. Az egyrétegű alkalmazás alapvetően egy olyan program, ami a felhasználó gépén fut. A három alapvető szolgáltatás (prezentáció, üzleti logika, adatkezelés) szerkezetileg egyetlen programba van beágyazva. Ezt ma már ritkán használják. A kétrétegű kliens/szerver alkalmazási modellnél az adatkezelést leválasztották, és egy adatbázis szerverre bízták. A háromrétegű kliens/szerver alkalmazási modellnél mind a három szolgáltatást különválasztották. Az üzleti logika szolgáltatásait egy alkalmazás szerverre telepítik. A négyrétegű kliens/szerver alkalmazásmodellnél egy Web alkalmazás esetén az alkalmazás szerver elkülönülhet a Web szervertől. A szolgáltatásoknak az ilyen formában való szétosztásával csökkenteni tudják az egyes komponensek közötti adatforgalmat, illetve a biztonságot tudják emelni.

8.1.2. Vékony kliens és vastag kliens megoldások

A Web alkalmazás által megvalósítandó funkciók szétbontása felhasználói interfész, üzleti logika, és adatkezelési szolgáltatásokra többféleképpen is megtörténhet.

A vékony kliens (thin-client) megoldásnál a funkciók lehető legkisebb részét valósítja meg a felhasználó gépén futó böngészőprogram. A munka jelentős részét a Web szerver, alkalmazás szerver, illetve az adatbázis szerver végzi. Előnye az, hogy a Web alkalmazás független a felhasználó gépének paramétereitől, nagyobb biztonsági szint érhető el. A hátránya abból adódik, hogy a böngésző kevés funkcióval van ellátva, így csak statikus HTML oldalak megjelenítésére alkalmas. Ez a mai felhasználók igényeit már nem elégíti ki.

További hátránya, hogy mivel mindent a funkciógazdag szervereknek kell elvégezni, ezért jelentős a szerverek és a hálózat terhelése.

A vastag kliens (thick-client) architektúra ennek pont a fordítottja. Igyekeznek minél több funkciót a kliens oldali böngészőkben megvalósítani. Ezért a dinamikus HTML kezelés is beépül a böngészőkbe. Képesek scripteket futtatni (JavaScript), ActiveX elemekkel bővítik a böngészők képességét. A megoldás előnye, hogy változatosak, dinamikusak lesznek az oldalak. Csökken a hálózat terhelése. A hátrányai, hogy a scriptek, vezérlők biztonsági réseket nyitnak, és az alkalmazások függővé válnak a kliens oldali szoftverektől. Ezért manapság a két véglet között próbálnak egyensúlyozni.

8.2. Adobe Dreamweaver

Szinte szabvánnyá vált HTML szerkesztő és webalkalmazás-fejlesztő eszköz. Képes arra, hogy formázott szöveget, képeket, kereteket, táblázatokat, űrlapokat hozzunk létre esztétikus formában, viszonylag rövid időn belül. Erőssége a Dynamic HTML (Dinamikus HTML) – DHTML, amely lehetővé teszi a felhasználók és a weboldal közötti párbeszédet. A beépített program kódjaival könnyen tudunk viselkedést rendelni az oldalainkhoz. Támogatja több programozási/formázó nyelv használatát (PHP, JavaScript, ASP, ColdFusion, XHTML, XML, XSLT). Kiváló támogatást ad a CSS-el történő szöveg formázására, illetve az oldal külalakjának kialakítására. Tiszta XHTML kódot készíthetünk vele, és tartalmazza a Spry keretrendszer viselkedéseit. Kezeli és karbantartja a létrehozott webhelyeket. Támogatja az újrahasznosítható könyvtárelemek és sablonok létrehozását. A weblapok tervezése során a gyakran használt objektumokból könyvtárelemeket készíthetünk. Később, ha az eredeti könyvtárelemet frissítjük, akkor az összes arra épülő példány követi a változást. Könyvtárelemet készíthetünk például navigációs sávokból. A sablonok használatával pedig biztosíthatjuk a webhelyünk egységes kinézetét, és könnyű kezelhetőségét. A sablon tulajdonképpen a tartalom és a megjelenés együttese.

8.3. XHTML (Extensible Hypertext Markup Language – Bővíthető Hipertext Jelölőnyelv)

Egy olyan formázó nyelv, amely azt írja le, hogy milyen szintaktikával kell megírni a weblapokat ahhoz, hogy a böngészőprogramok a megfelelő módon jelenítsék meg. A HTML

nyelvből fejlesztették ki. Az XHTML az Extensible Markup Language (XML) nyelvtani szabályait követi, amelyek szigorúbbak a HTML nyelvénél. A bővíthetőség arra utal, hogy különböző modulokkal bővítve, olyan feladatok végrehajtására lesz alkalmas, mint például a képek megrajzolása, vagy matematikai számítások elvégzése.

Az XHTML nyelvre vonatkozó fontosabb szabályok:

- Minden weblapot el kell látni dokumentumtípus (DOKTYPE) deklarációval.
- Minden elemet, jellemzőt és értéket kisbetűvel kell írni.
- Minden értéket idézőjelben (', ") kell megadni.
- Minden jellemzőnek egyértelműen kell megadni az értékét.

8.4. CSS (Cascading Style Sheets – Rangsorolt Stíluslapok)

Arra használhatjuk, hogy megjelenítést rendeljünk a szerkezetet megadó XHTML/HTML kódhoz. Segítségével beállíthatjuk a weblapokat alkotó elemek elrendezését, szövegek, objektumok megjelenését. A stílust meghatározó leírást elhelyezhetjük az XHTML dokumentumban, illetve külső fájlban. Abban az esetben, ha külső stíluslapot használunk, több weblap is használhatja a CSS stílust. Így azonos megjelenést érünk el, könnyebb a karbantartás, illetve a közös használat miatt csak egyszer töltődik le.

A stílusokat kijelölők (selector), illetve meghatározások (declaration) segítségével lehet megadni. A kijelölők a weboldal valamelyik elemét hivatkozzák. Ilyen például a „body" dokumentum törzs kijelölő. A meghatározások pedig megadják, hogy a kijelölt elem hogyan jelenjen meg a megjelenítő eszközön (képernyő, nyomtató, kivetítő, stb.).

```
body {
    background-color: #000000;
}
```

A „background-color: #000000" meghatározással fekete háttérszínt állítottunk be.

8.5. PHP (PHP: Hypertext Preprocessor)

Kezdetben egy makrókészlet volt, amely személyes honlapok karbantartására készült. Később jelentősen bővült a képessége és egy önállóan használható programozási nyelvvé fejlődött. Tulajdonképpen egy kiszolgáló oldali parancsnyelv, amit jellemzően HTML oldalakon használnak. A PHP parancsokat a kiszolgáló oldalon dolgozza fel a PHP-értelmező motor és egy előállított HTML lapot küld vissza az ügyfélnek. A szintaktikája hasonlít a C-re, a nyílt forráskód miatt ingyenes és sok fejlesztést támogató könyvtárat tartalmaz. Kiemelt együttműködést biztosít az adatbázisokhoz, többek között az ingyenes MySQL-hez is. A PHP 5-ös verziójába már beépítették az XML támogatást.

8.6. MySQL adatbáziskezelő-rendszer (ABKR)

A MySQL az egyik legelterjedtebb adatbáziskezelő-rendszere, amely a relációs modellre épül. Tárolja a tábláink szerkezetét, a közöttük fennálló kapcsolatokat, a táblákban lévő adatokat, lehetővé teszi a felhasználók és a hozzájuk tartozó jogosultságok megadását, naplózza az egyes felhasználók tevékenységeit, és lekérdezésekkel teszi lehetővé az adatok kinyerését. A lekérdezés egy olyan SQL nyelven írt utasítássorozat, amelyet az ABKR feldolgoz, és a feldolgozás eredményeként kapott adatokat eredménytábla formájában jeleníti meg. SQL = Structured Query Language (strukturált lekérdező nyelv). A relációs adatbázis-kezelés szabványos nyelve. A MySQL támogatja a többszálú végrehajtást, ami azt jelenti, hogy minden egyes új kapcsolat létesítésekor elindul egy új kiszolgálófolyamat. Ezért megfelelő teljesítményt nyújt. Még egy jelentős érv a MySQL mellett az, hogy ingyenes. A szolgáltatásait elérhetjük például a phpMyAdmin grafikus felületű szoftverrel, és PHP script-ből egyaránt.

8.7. JavaScript

A Netscape által létrehozott parancsnyelv, amely LiveScript néven jelent meg 1995-ben a Netscape Navigator 2.0-ban lévő értelmezőmotorral. Mára a legelterjedtebb webes parancsnyelv. Többnyire kliensoldali programokban használják. Az egyik legfontosabb eszköze a DOM, azaz dokumentumobjektum-modell, amelynek segítségével weboldalakat, ablakokat, dokumentumokat kezelhetünk. A DOM nem része a nyelvnek, az egy API (alkalmazásprogramozási interfész), amelyeket a böngészőkbe építenek be. A DOM

objektumai tulajdonságokkal és metódusokkal rendelkeznek, amelyeket JavaScript-ből kezelhetünk. Az objektumok hierarchikusan szervezettek. A window.document.forms[] hivatkozás egy objektum szerkezetet is meghatároz. A „window” az objektumok szülője, a böngészőablakot képviseli. Annak a gyermeke a „dokument”, amellyel a weboldalt tudjuk hivatkozni. A „forms[]” pedig a weboldalakon elhelyezett űrlapok tömbje.

9. Kódolás

9.1. Webhely, weblapok létrehozása

A webhely létrehozására az Adobe Dreamweaver CS3 programját használom. Mivel a fejlesztés a helyi lemezen történik, ezért első lépésként meg kell határozni a főkönyvtárat. Ebben a könyvtárban lesznek azok a fájlok, amelyeket a webhelyre is fel fogok tölteni. A webhelyek meghatározásánál a Dreamweaver ezt a könyvtárat és a benne szereplő fájlokat tekinti az adott webhely tárolójának, és a belső hivatkozásokat ehhez fogja beállítani. A Web szerver könyvtár hierarchiájában lesz egy mappa létrehozva, amely a webhelyünk gyökér könyvtáraként fog funkcionálni.

A helyi lemezen a következőképpen alakítottam ki a webhely lokális mappaszerkezetét

web mappa: Minden az adott webhellyel kapcsolatos fájlt ide helyezek el.

- adatbazis mappa: Az adatbázis-kezeléssel kapcsolatos fájlok.
  - ab_kepek mappa: Az adatbázissal kapcsolatos képek.
  - Az elméleti és gyakorlati fejezetek weblapjai.
  - A php kiterjesztésű tesztlap.
- files mappa: A felhasználók által letölthető kiegészítő fájlok (pl. txt).
- kepek mappa: A teljes webhelyhez tartozó képek.
- scripts mappa: A css, php, javascript fájlok.
- templates mappa: A weboldalak alapját képező sablon fájl.
- index.html: A webhely nyitó oldala.

Az általam használt weblapok szerkezetileg hasonlítanak egymáshoz, ezért egy általánosan használható sablont készítettem. Ennek segítségével biztosítható, hogy a webhelyünk egységes képet mutasson és könnyen kezelhető legyen. A sablonokba definiálhatunk szerkeszthető és nem szerkeszthető területeket. A módosításra vonatkozó korlátozások a sablon alapján létrehozott weblapokat érintik. A nem módosíthatóra bejelölt

területeket, ezekben a weblapokban nem engedi módosítani a Dreamweaver. Viszont, ha a sablonon változtatunk, akkor az érvényre jut valamennyi ebből létrehozott oldalon. A szerkeszthető területek pedig szabadon megváltoztathatjuk.

Alapértelmezésben a Dreamweaver a sablon egész területét zárolja, ezért nekünk kell meghatározni a szerkeszthető területeket az Insert/Template Objects/Editable Region menüponttal.

A következő sablont hoztam létre:

3. ábra: A basic.dwt sablon fájl

A felső navigációs rész, a tartalom terület és az alsó navigációs rész szerkeszthetővé lett téve.

A sablonhoz a következő bejegyzéseket rendeltem:

- Az xhtml dokumentum-típus deklarációja:
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

- A karakterkészletet definiáló meta tag:
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

- A külső stíluslap összekapcsolása az xhtml oldallal, egy hivatkozáscímkén keresztül:
  <link href="../scripts/basic.css" rel="stylesheet" type="text/css">

A sablon alapján új weblapot a Dreamweaver File/New/Page from Template parancsával lehet készíteni. Ha kiválasztjuk az Update page when template changes választónégyzetet, akkor folyamatos kapcsolatot teremtünk a sablon és a rá épülő weblapok között, így ez utóbbi követi a sablon változásait.

A basic.dwt sablon alapján készítettem el a következő xhtml oldalakat:

- index.html: A webhely kezdő oldala. Megkönnyíthetjük a felhasználók útját, ha a webhelyen az „index.html”, vagy „default.html” nevet adjuk a kezdőlapunknak. Ebben az esetben, ha a felhasználó nem ad meg fájlnevet elérési úttal, akkor a böngészőprogramok automatikusan az ilyen nevű oldalakat keresik.
- ab_tartalom.html: Az adatbázis elmélet tartalomjegyzéke.
- ab_elso.html – ab_kilencedik.html: Az adatbázis elmélet oldalai.
- ab_gy_tartalom.html: Az adatbázis gyakorlat tartalomjegyzéke.
- ab_gy_elso_hetedik.html – ab_gy_kilencedik.html: Az adatbázis gyakorlat oldalai.
- ab_teszt.php: Az adatbázis teszt oldala.

Érdemes elválasztani a weblapok szerkezetét a megjelenésétől. A különválasztás lehetővé teszi, hogy a HTML nyelvet csak a szerkezet kialakítására használjuk. Ezzel sokkal átláthatóbb lesz kód, és a formázó utasítások nem növelik letöltésnél a weblapok méretét. Egy további előny, hogy ugyanahhoz a szerkezethez többféle formázás is megadható. Például külön elkészítjük a megjelenést normál monitoros felületre, a kisebb kijelzőjű mobiltelefonokhoz vagy papírra való nyomtatáshoz.

9.2. A menü szerkezet és formátum kialakítása

A menü szerkezetét, vagyis HTML kódját egy lista szerkezet adja. Ezzel elérhetjük, hogy nem csak logikailag, hanem szerkezetileg is együtt vannak az összetartozó elemek. Nincs

szükség külön látványelemek, képek vagy JavaScript használatára, mert a kialakítás megoldható a CSS stílusnyelv segítségével. A képek feleslegesen növelnék az átvitt adat mennyiségét, a JavaScript futtatására nem képes felhasználók pedig búcsút mondhatnának a dizájnos megjelenésnek.

Nézzük a sablon főmenü szerkezetét adó HTML kódot:

```
<div id="fo">
   <ul>
         <li><a href="#">Home</a></li>
         <li><a href="#">Adatbázis</a></li>
         <li><a href="#">Programozás</a></li>
         <li><a href="#">Hardver</a></li>
   </ul>
</div>
```

Az <ul> html tag-el számozás nélküli listát hozhatunk létre (UL – unordered list), amely alapértelmezett vagy beállított felsorolásjeleket használ a listaelemek jelölésére. Mind a számozott és a számozás nélküli listákban az <li> tag a listaelemeket jelöli (LI – list item). Egyik weblapról egy másikra, vagy ugyanazon weblap más pontjára úgynevezett horgonyok (anchor) segítségével tudunk eljutni, amelyet az <a> html tag képvisel. Az <a> tag „href” jellemzőjében kell megadni a hivatkozott célt. A tag egy blokk képző elem, amelynek segítségével a tartalom területét a CSS dobozmodelljének megfelelően tudjuk formázni. A tag „id” jellemzője azonosítja az adott elemet, így ehhez a területhez formázó utasításokat tudunk rendelni.

9.3. A menü formázása CSS segítségével:

Első lépésben beállítottam annak a blokknak a tulajdonságait, amelyikben az egész fő navigációs lista elhelyezkedik. A # jel segítségével a HTML elemeket az azonosító „id” tulajdonságaikon keresztül tudom hivatkozni. Ilyen formában a „#container #navig #fo”

azonosító sorozattal „container” elemet, majd azon belül a „navig” elemet és végül a „fo” azonosítóval rendelkező HTML tag-et érem el. Egy azonosítóból az egész HTML dokumentumban csak egy szerepelhet. De a gyakorlatban úgy vettem észre, hogy több elemhez is hozzá lehet rendelni ugyanazt az azonosítót.

A „background-color” tulajdonsággal megadom a blokk háttérszínét, A „padding” segítségével a CSS dobozmodellnek megfelelően beállítom a tartalom és a szegély közötti távolságot. A „border” jellemzőkkel a felső (top), illetve az alsó (bottom) szegély szélességét (width), stílusát (style), színét (color).

```
#container #navig #fo {
    background-color: #4d7096;
    padding: 0.1em;
    border-top-width: medium;
    border-bottom-width: medium;
    border-top-style: solid;
    border-top-color: #990000;
    border-bottom-style: solid;
    border-bottom-color: #990000;
}
```

A következő utasítással a „#container #navig #fo” azonosító úttal jelölt „ul” HTML tag-et érem el. Így az ebben a blokkban szereplő formázó utasítások csak a kijelölt általános HTML elemre lesznek alkalmazva. Itt csak a szöveg igazítását (text-align) állítom középre (center).

```
#container #navig #fo ul {
    text-align: center;
}
```

Az egysoros menüszerkezet kialakítását a listaelemeken „li” végzem el. Ezek az utasítások is csak a „container” – „navig” – „fo” azonosítókkal jelölt blokkokon belüli „li” HTML tag-re érvényes. A „display: inline” jellemző: érték párossal a listaelemek soron belül, egymás utáni elhelyezkedését adom meg. A „list-stile-type: none” pedig a felsorolásjelek eltávolítását végzi. A „margin-right”, „margin-left” jobb és baloldalról 10 pontos térközt adnak a menüelemeknek.

```
#container #navig #fo li {
    display: inline;
    margin-right: 10px;
    margin-left: 10px;
    list-style-type: none;
}
```

A fő navigációs rész horgonypontjainak aláhúzását a „text-decoration: none” utasítással távolíthatjuk el. Beállítom a háttérszínt (background-color), a betűszínt (color), a keret (border) vékony voltát (thin), stílusát (solid) és színét. A „letter-spacing: 0.1 em” értékkel kis betűközt állítok be.

```
#container #navig #fo a {
    text-decoration: none;
    padding-top: 0.15em;
    padding-right: 0.5em;
    padding-bottom: 0.15em;
    padding-left: 0.5em;
    background-color: #0000CC;
    color: #000000;
    border: thin solid #000066;
```

```
    font-size: 24px;
    font-weight: bold;
    letter-spacing: 0.1em;
}
```

A hivatkozások állapot szerinti formázására a CSS „álosztály-kijelölőket” használ. A hivatkozások azért álosztályok, mert hivatkozás állapota nem szerepel az XHTML kódban, ez a felhasználó tevékenységétől függ. A négy leggyakoribb álosztály a következő:

- a: link, (hivatkozás)
- a: visited, (felkeresve)
- a: hover, (fölötte állva)
- a: active (használatban).

A felkeresve és a fölötte állva állapotok színeit változtattam meg.

```
#container  #navig  #fo a:visited {
    background-color: #000099;
}
#container  #navig  #fo a:hover {
    color: #990000;
}
```

Az így megformázott egyik menü:

4. ábra: Fő navigációs sáv

9.4. A relációs adatmodell meghatározása

Az adatok tárolására egy konkrét adatbáziskezelő-rendszert (ABKR) kell használni. Én a MySQL relációs adatbáziskezelő-rendszert választottam. A fent elkészített Egyed – Kapcsolat modellt ezért át kell alakítani relációs modellé. A relációs adatmodellt 1970-ben definiálta E. F. Codd amerikai kutató, de gyakorlati alkalmazása csak az 1980-as években vált általánossá. Lényege, hogy az egyedeket, tulajdonságokat és kapcsolatokat egyaránt táblázatok, úgynevezett adattáblák (relációk) segítségével kezeli. A relációs adatmodellnél a tulajdonságok kapják a fő hangsúlyt, a tulajdonságokkal definiáljuk az adatmodell szerkezetét. A relációs modellben a kapcsolatokat a kulcs – kapcsoló kulcs mezők értékeinek egyező volta hozza létre.

Első lépés a relációs sémák megtervezése:

- feladatok (fazon, kategotia, fleiras, helyes)
- valaszok (vazon, vnev, vleiras, fazon)

Relációs sémának nevezünk egy attribútum halmazt, amelyhez azonosító nevet rendelünk.

A „feladatok” reláció tulajdonságainak leírása:

- fazon → feladat azonosítója,
- kategoria → feladat kategória,
- fleiras → feladat leírás,
- helyes → helyes válasz

Az „fazon” tulajdonság azonosító jellegű, vagyis minden rekordnál különböző értéket kell felvennie. A többi tulajdonság leíró jellegű.

A „valaszok” reláció tulajdonságainak leírása:

- vazon → válasz azonosító,
- vnev → válasz megnevezése,

- vleiras → válasz leírása
- fazon → feladat azonosítója.

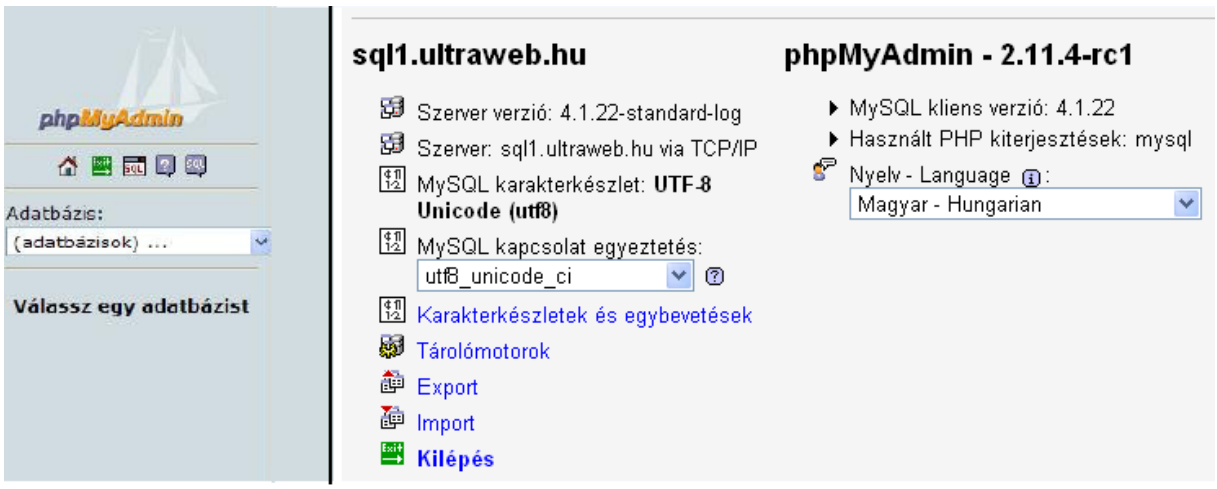
A „vazon” jellemző azonosító jellegű, a „vnev”, „vleiras” pedig leíró tulajdonságok. Az „fazon” egy speciális szerepben van, úgynevezett külső kulcsként a kapcsolatot biztosítja a „feladatok” táblával. Mivel a „feladatok” és a „valaszok” relációk között egy:több típusú kapcsolat van, - ugyanis egy feladathoz több válasz is tartozhat – ezért a „valaszok” relációban több rekordnak is lehet ugyanolyan értéke. A kulcsot a relációs sémában folytonos aláhúzással, a külső kulcsot szaggatott vonalas aláhúzással jelőljük.

A „feladatok” reláció „helyes” tulajdonsága, és a „valaszok” reláció „vnev” jellemzőjének értékei közös halmazból kerülnek ki, azaz domain-jük megegyezik.

Dom(feladatok.helyes) = Dom(valaszok.vnev) = {’a’, ’b’, ’c’, ’d’}

9.5. A MySQL adatbázis elkészítése

A phpMyAdmin meghívásakor először elkéri a felhasználói nevünket és a jelszavunkat. Sikeres belépés esetén a kezdő oldalon találjuk magunkat.



5. ábra: phpMyAdmin bejelentkező képernyője

Mivel egy ingyenes web szolgáltatót használok, a regisztrálás után létrejön egy adatbázis, amiben tetszőleges számú táblát hozhatunk létre. Tehát az adatbázis adott, így a következő lépés az adatbázist alkotó táblák szerkezetének meghatározása. Ezt elvégezhetjük a phpMyAdmin tábla-tervezőjével, vagy az SQL nyelv segítségével.

A táblákat létrehozó SQL parancsok:

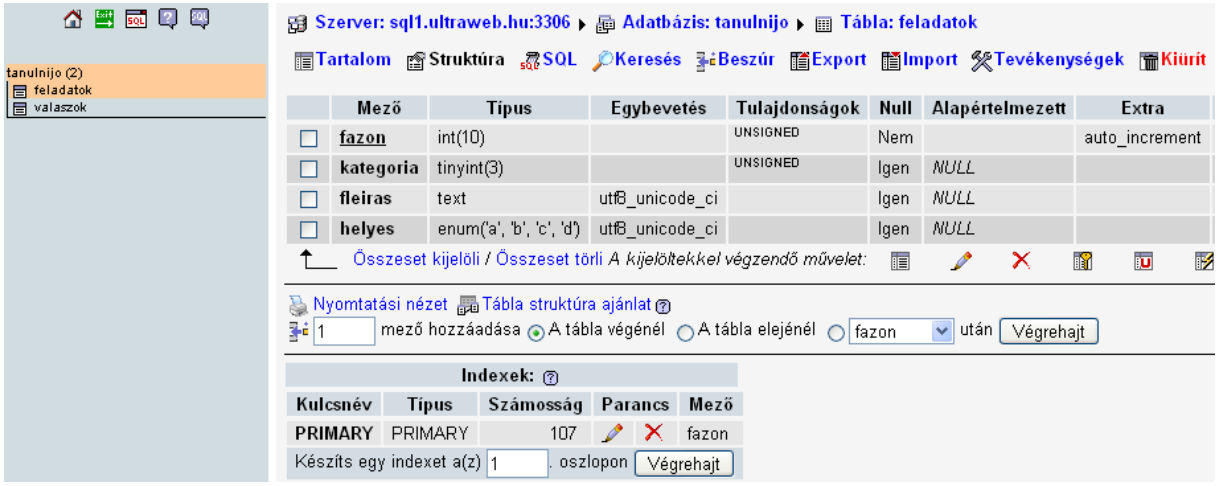
```
CREATE TABLE feladatok (
  fazon INT(10) UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
  kategoria TINIINT(3)UNSIGNED
  fleiras TEXT
  helyes ENUM(’a’, ’b’, ’c’, ’d’)
);

CREATE TABLE valaszok (
  vazon INT(10) UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
  vnev ENUM(’a’, ’b’, ’c’, ’d’)
  vleiras TEXT
  fazon INT(10) UNSIGNED
  FOREIGN KEY (fazon)
  REFERENCES feladatok(fazon)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

A „fazon”, „vazon” azonosító, automatikusan növekvő egész adattípust kapott. Ez azt jelenti, hogy új rekord felvitelénél a felhasználónak nem kell megadni az értékét, a rendszer automatikusan egyel növeli. A „helyes” és a „vnev” felsorolt típus.

A „valaszok” tábla ’fazon” tulajdonsága külső kulcs (FOREIGN KEY), amivel a „feladatok” tábla „fazon” kulcsára (REFERENCES) hivatkozik. A két megszorítással (ON DELETE/UPDATE CASCADE) elérjük, hogy a „valaszok” táblából töröljön minden olyan sort, amelyhez nem tartozik feladat, illetve a feladat azonosítójának megváltozása esetén a hozzá tartozó válaszokban is elvégezze a módosítást.

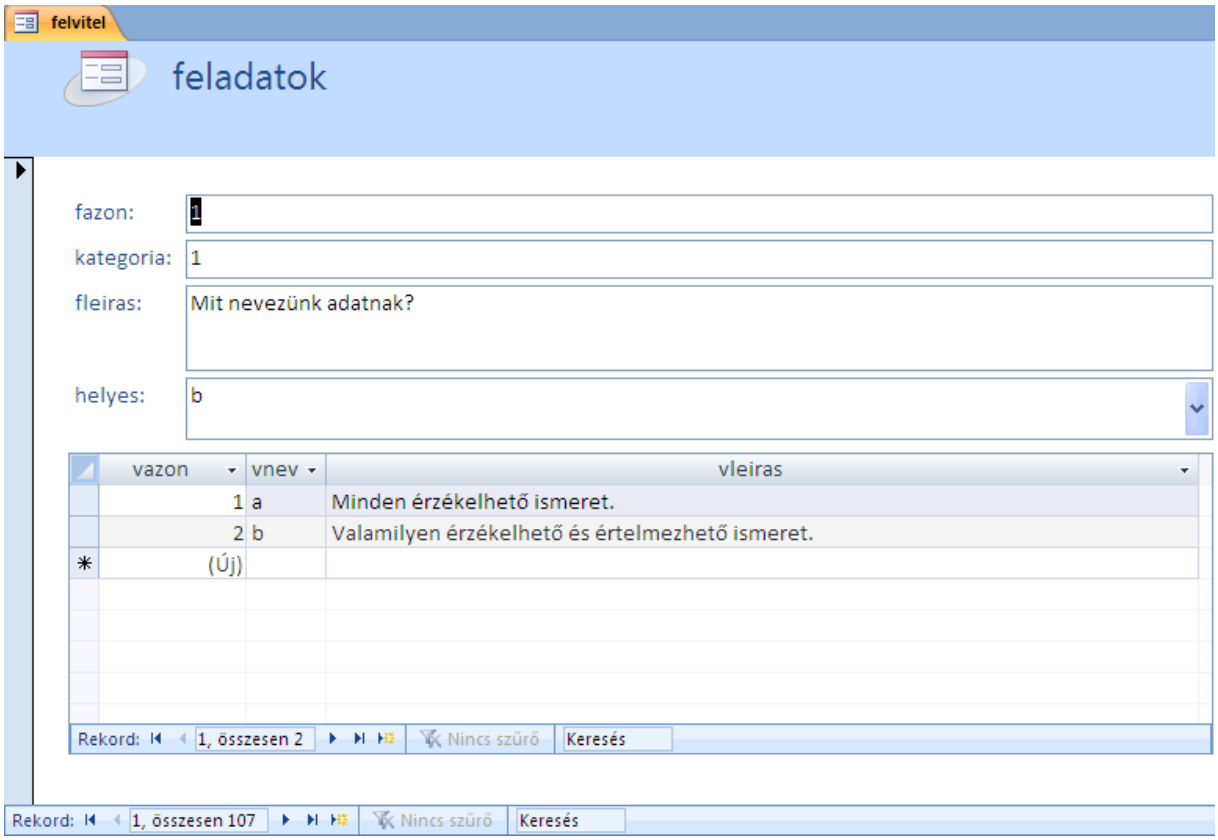
A phpMyAdmin-ban a „feladatok” tábla szerkezete a következő:



6. ábra: A „feladatok” tábla szerkezete

Ugyanígy jelenik meg a „valaszok” tábla szerkezete is. A karakterkódolás miatt fontos, hogy a szöveges mezők karakterkódját egységesen kezeljük a weblapokon, az adatbázisban és a majdani PHP kódokban is. Én UTF-8 karakterkódolást választottam.

A táblaszerkezetek létrehozása után következik az adatokkal való feltöltésük. A feltöltést elvégezhettem volna a phpMyAdmin felületén rekordonként, de az túl hosszadalmas lett volna. Ezért készítettem egy Access adatbázist egy adatfelviteli űrlappal, és azt használtam az adatok felvitelére.



7. ábra: Access adatfelviteli űrlap

Miután felvittem az adatokat, exportáltam két UTF-8 kódolású szövegfájlba (feladatok.txt, valaszok.txt), az elválasztó karakternek pontosvesszőt állítottam be. A MySQL-ben pedig importáltam a megfelelő táblákba. Erre a LOAD DATA INFILE parancs használható:

```
LOAD DATA INFILE 'feladatok.txt'
INTO TABLE feladatok
FIELDS TERMINATED BY ';';

LOAD DATA INFILE 'valaszok.txt'
INTO TABLE valaszok
FIELDS TERMINATED BY ';';
```

A szövegfájlokban a mezők elhatárolására a pontosvesszőt használtam, így ezt meg kellett adni a LOAD DATA INFILE parancsban is a FIELDS TERMINATED BY utasítást követően.

A feltöltött „feladatok” tábla részlete a phpMyAdmin felületen keresztül:

	fazon	kategoria	fleiras	helyes
	1	1	Mit nevezünk adatnak?	b
	2	1	Mit nevezünk adatnak?	a
	3	1	Mire valók az adatok?	b
	4	1	Amikor egy autókereskedés nyílvántartásába autó ad...	c
	5	1	Mit nevezünk információnak?	b
	6	1	Van-e különbség az adatfeldolgozás és az informáci...	c
	7	1	Mivel foglalkozott Shannon?	b
	8	1	Shannon szerint melyik állításnak van nagyobb info...	a
	9	1	Amikor adatokból információt szeretnénk kinyerni, ...	b
	10	1	A következő autót jellemző leírás adat, vagy infor...	a
	11	2	Mivel foglalkozik az adatkezelés?	b
	12	2	Miért végzünk adatkezelést?	c
	13	2	Az adatkezelésnek hányféle formáját ismerjük?	a
	14	2	Milyen az adatszerű adatkezelés?	b

8. ábra: feltöltött „feladatok” tábla

9.6. A tesztlap (ab_teszt.php) elkészítése

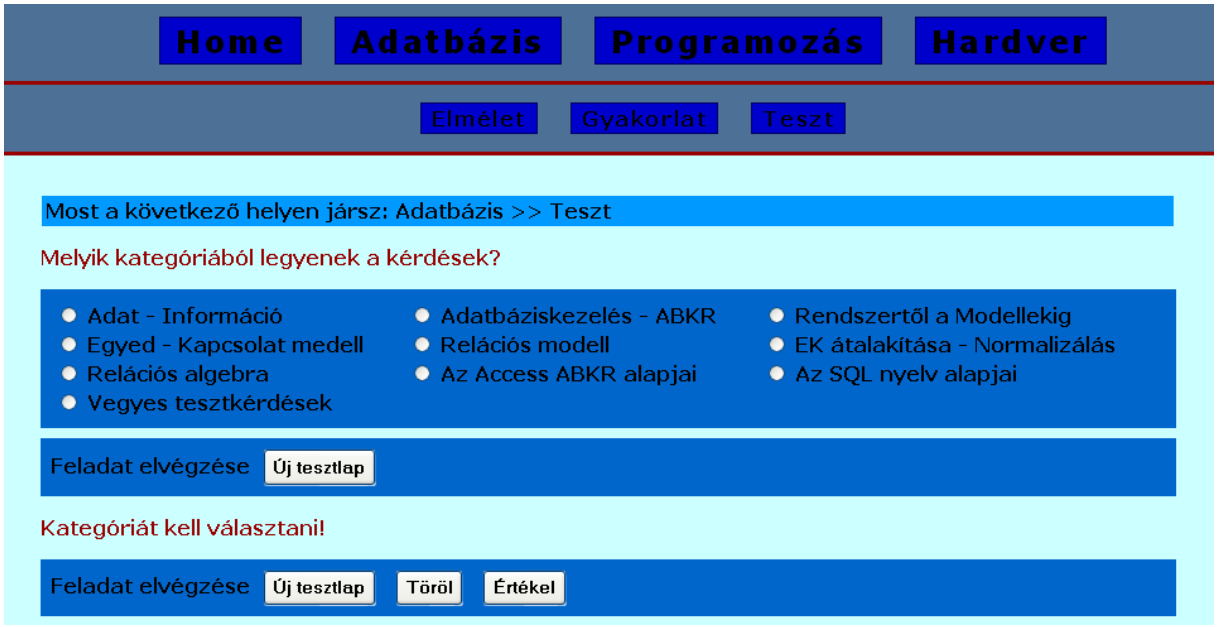
A tesztlap egy részben dinamikusan előállított oldal, amelyet úgy alakítottam ki, hogy sablon alapján létrehoztam egy kiindulási vázat, a feladatokat pedig egy PHP script a felhasználó választásának megfelelően állítja elő. A feladatok a kategóriákon belül véletlenszerűen lesznek kiválasztva. Az egyes kategóriákból ötösével, az egész anyagból pedig tízesével.

Az oldal felhasználói esetei:

- kategória választás: A felsorolt kategóriák közül választani kell és annak megfelelő kérdések lesznek generálva.
- új tesztlap kérés: Egy nyomógombbal kiváltott esemény, amely a kategória választásnak megfelelően a szerverre továbbítja a tesztlap összeállításának a kérését.
- törlés: Törli a tesztlap eddig kiválasztott feladat megoldásait.

- kiértékelés: Ellenőrzi a felhasználó válaszait. Egy kliensoldali script végzi, hogy kevesebb legyen a szerverhez fordulás.

A bejelentkező tesztoldal képe:



9. ábra: A tesztoldal választás előtt

A kategória választás után az „Új tesztlap” gombbal lehet lekérni a feladatokat.

10. ábra: A tesztoldal feladatokkal

A tesztoldal felhasználói interaktivitása miatt űrlapot, és rajta elhelyezett vezérlőket tartalmaz. Az űrlap (FORM) rendelkezik néhány jellemzővel, amelyek kitöltése függ a használat módjától. Az ab_teszt.php oldalon a következő alakú:

<form action="<?php print $SERVER['PHP_SELF']?>" method="post" enctype="multipart/form-data" name="ab_form" id="ab_form"> … </form>

Az action (művelet) jellemző az űrlap adatait feldolgozó parancsfájlra mutat. Ez lehet külön állományban, vagy kis programoknál a form-ot tartalmazó weboldalon is. A $SERVER['PHP_SELF'] szerkezet, egy globális változó, amely a feldolgozó program helyének az aktuális fájlt jelöli meg. A method (módszer) jellemző az elküldés módját adja meg, értéke lehet „post”, vagy „get”. A „post” több adat elküldését teszi lehetővé, és az elküldés módja is biztonságosabb. Az enctype (kódolás) tulajdonság az űrlapadatok MIME kódolási típusát tartalmazza.

Az űrlapon a felhasználó ténykedéseit vezérlőelemek fogadják, amelyek a következők lehetnek:

- szövegbeviteli mező (TEXT),
- többsoros szövegmező (TEXTAREA),
- választólista (SELECT),
- választónégyzet (CHECKBOX),
- rádiógomb (RADIO),
- parancsgomb (BUTTON)

A tesztoldal előállítását egy szerveroldalon futó PHP nyelven megírt script végzi. A programot egy külön fájlban készítettem el, amelyet az INCLUDE() függvénnyel építek be a weboldalba. A külső fájlban szereplő PHP kód úgy hajtódik végre, mintha a dokumentum része lenne.

A beágyazást végző programkód:

```
<?php
    include_once("../scripts/teszt.php");
?>
```

A feldolgozást végző programkód az 1. számú mellékletben található.

Az adatbázisból a következő SQL lekérdezésekkel kapom meg a szükséges adatokat. Előbb lekérdezem a felhasználói választásnak megfelelő kategóriához tartozó rekordokat:

```
$f_lekerdezes = "SELECT fazon, fleiras, helyes FROM feladatok
WHERE kategoria = '$kategoria'";
```

A kapott eredménytáblát asszociatív tömbként tudom kezelni, és a $f_lekerdezes változóval hivatkozom rá. Majd ebből a lekérdezésből választok ki véletlenszerűen rekordokat, és megjelenítem feladatonként külön-külön, a hozzá tartozó válaszokkal. A válaszokra hivatkozni rádiógombokkal lehet.

```
print "<label><input type=\"radio\"
   name=\"RadioGroup_v".$feladat_sorszam."\"
   id=\"rg_v".$feladat_sorszam."\"
   value=\"".$v_sor['vnev']."\">".$v_sor['vleiras']."
   </label><br>";
```

9.7. A tesztlap kiértékelésének megvalósítása

A kitöltés helyességét egy JavaScript programmal végzem. Így az ellenőrzés nem terheli a hálózatot, viszont megvan az a hátránya, hogy a tesztlap összeállításakor le kell küldeni a feladatok helyes megoldásainak értékét is, rejtett mezőkben.

```
print "<input type=\"hidden\" name=\"f_".$feladat_sorszam."\"
id=\"f_".$feladat_sorszam."\"
value=\"".mysql_result($f_eredmeny, $szam-1, 'helyes')."\"/>";
```

A rejtett (hidden) tulajdonságérték miatt a mező nem fog látszani a böngészőkben. A „value” tulajdonság pedig a feladathoz tartozó helyes értéket kapja.

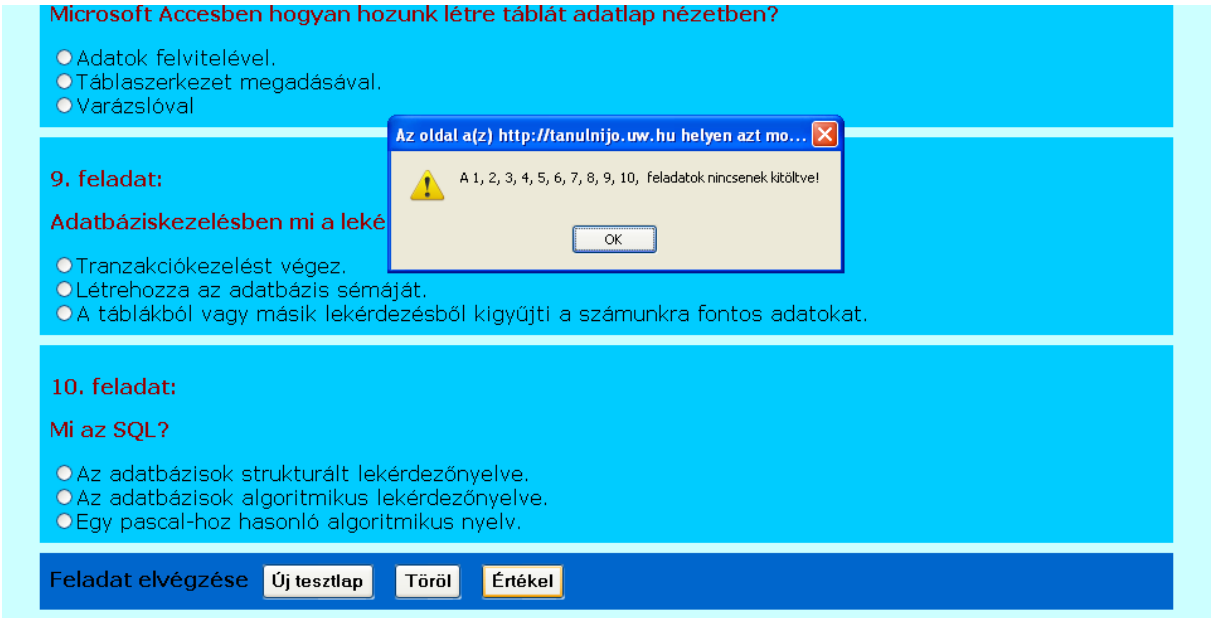
Az értékelés indítását egy parancsgomb lenyomása végzi. A „Kiértékel” gomb „onClick” eseményéhez rendelt „ertekel()” függvény fog lefutni.

```
<input type="button" name="btn_ertekel" id="btn_ertekel"
class="ab_teszt_gomb" value="Értékel" onClick="ertekel();">
```

A kiértékelést végző program elvi felépítése:

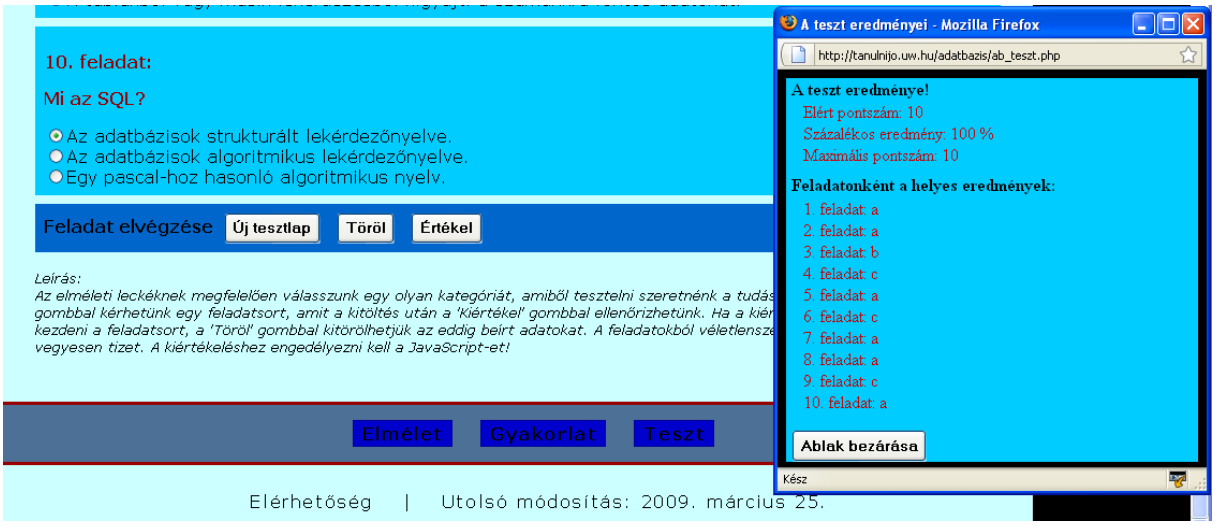
- Ellenőrzi, hogy minden feladatnál volt-e választás. Abban az esetben, ha nincs kiválasztva valamelyik feladatnál rádiógomb, tájékoztatja a felhasználót arról, hogy melyik feladatot nem oldotta meg.
- Meghatározza a maximális pontot.
- Majd egy ciklusban feladatonként végigmegy és összehasonlítja a kitöltött értéket a helyessel és, ha egyezés van növeli az elért pontot. Egyúttal elkészíti a helyes eredményeket tartalmazó tömböt is.
- A ciklusból kilépve kiszámítjuk az elért százalékot, majd egy ablakban jelenítjük meg az eredményt és a helyes eredményeket is.

Ha nem válaszolunk minden feladatra, a következő üzenetet kapjuk:



11. ábra: Kiértékelés hibaüzenete

A kiértékelés eredmény ablaka a következő:



12. ábra: Kiértékelés eredmény ablaka

A program forráskódja a 2. mellékletben található.

10. Tesztelés

A fejlesztés során folyamatos tesztelést alkalmaztam. Ahogy elkészült egy modul, feltöltöttem a szerverre, és ellenőriztem különböző böngészőkben. A tesztelés során jelentős segítséget adott az Adobe Dreamweaver szintaktikai kiemelése, és Code/Design nézete. Ezen kívül a web fejlesztők tökéletes partnere a Mozilla Firefox böngésző és a hozzá elérhető fejlesztőt támogató kiegészítések. A következő kiegészítéseket használtam:

- web-developer: Fejlesztéshez hasznos eszköztárat ad.
- colorzilla: Színelemzés a böngészőben.
- firebug: Lehetőséget nyújt HTML-analízisre, CSS-hibakeresésre és layoutok szerkesztésére, DOM-elemzésre és JavaScriptek debuggolására.

A kiterjesztések letölthetők a következő címről: www.firefox.hu

11. Dokumentálás

A dokumentálásnál két szempontot kell figyelembe venni:

- A felhasználók felé történő dokumentáció: Úgy gondolom, hogy a webhely elég kicsi és egyszerű ahhoz, hogy az oldalakra írt ismertető elegendő a helyes használathoz. Nem szükséges a funkciók ismertetéséhez külön oldalt létrehozni.
- A fejlesztői dokumentáció: A későbbi módosításokhoz a fejlesztő/programozó segítésére szokták létrehozni. A jelen leírás viszont kellő mélységű ahhoz, hogy a további fejlesztési/karbantartási feladatok könnyen megoldhatóak. Illetve a programkódot megjegyzésekkel láttam el, a könnyebb eligazodás végett.

12. Irodalomjegyzék

Magyar nyelvű szakirodalom:

- Betsy Bruce, Tanuljuk meg az Adobe Dreamweaver CS3 használatát, Kiskapu, 2007
- Virginia DeBolt, HTML és CSS Webszerkesztés stílusosan, Kiskapu, 2005
- Dave W. Mercer - Allan Kent - Steven D. Nowicki - David Mercer - Wankyu Choi, PHP5, Panem, 2006
- Matt Zandstra, Tanuljuk meg a PHP5 használatát, Kiskapu, 2005
- Julie C. Meloni, Tanuljuk meg a MySQL használatát, Kiskapu, 2003
- Michael Moncur, Tanuljuk meg a JavaScript használatát, Kiskapu, 2006
- Raffai Mária, Információrendszerek fejlesztése és menedzselése, Novadat, 2003
- Halassy Béla, Adatmodellezés, Nemzeti Tankönyvkiadó, 2002
- Kovács László, Adatbázisok tervezésének és kezelésének módszertana, Computerbooks, 2004
- Jeffrey D. Ullmann - Jenifer Widom, Adatbázisrendszerek, Panem, 1998

Magyar internetes forrás:

- http://weblabor.hu

Külföldi internetes forrás:

- JavaScript referencia - http://www.w3schools.com/jsref/default.asp
- PHP referencia - http://www.w3schools.com/PHP/DEfaULT.asP
- MySQL hivatalos weboldal - http://www.mysql.com/

13. Zárszó

Remélem az elkészült anyag hasznos segítséget fog jelenteni mindazok számára, akik középiskolai szinten szeretnének megismerkedni az adatbáziskezelés elméletével és gyakorlatával.

A későbbiekben szeretném elkészíteni a programozással és a hardverrel foglalkozó oldalakat A tesztoldalt pedig úgy bővíteni, hogy az osztályozást is lehetővé tegye.

A böngészőfüggetlenség megoldása még részben előttem áll. Úgy vettem észre, hogy ezt szakkönyvekből lehetetlen elsajátítani, inkább a hosszas kísérletezés fog célra vezetni.

14. Mellékletek

1. számú melléklet: A tesz.php fájl tartalma

```
<?php
//Hogy a feldolgozó is UTF-8 karakterkódolást használjon.
header('Content-type: text/html; charset=UTF-8');
//A függvény meghatározza, hogy egy szám benne van-e egy tömbben.
      function benne_van(&$sz, &$tomb){
            foreach($tomb as $elem) {
                  if($elem == $sz) {
                        return true;
                  }
            }
      return false;
      }
//Kapcsolódás a MySQL adatbázis-kiszolgálóhoz.
$kapcsolat = @mysql_connect("valahol", "azonosító", "jelszó");
//Ha sikertelen, hibakezelés.
if (!$kapcsolat) {
      die("Nem lehet kapcsolódni a MySQL kiszolgálóhoz! ".mysql_error());
}
//Az adatbázis kiválasztása, hibakezeléssel.
@mysql_select_db("tanulnijo") or
      die("Nem lehet megnyitni a tanulnijo adatbázist! ".mysql_error());
//Karakterkódolási paraméterek beállítása.
mysql_query("set names 'utf8'", $kapcsolat);
mysql_query("set character set 'utf8'", $kapcsolat);
//Megnézzük, hogy bejelölték-e valamelyik rádiógombot. Ha igen elkészítjük
a megfelelő lekérdezés szövegét.
if(isset($_POST['RadioGroup1'])) {
      $kategoria = $_POST['RadioGroup1'];
      if($kategoria != 10) {
            $f_lekerdezes = "SELECT fazon, fleiras, helyes FROM feladatok
WHERE kategoria = '$kategoria'";
            $feladatszam = 5;
            }
      else {
            $f_lekerdezes = "select fazon, fleiras, helyes from feladatok";
            $feladatszam = 10;
      }
//Végrehajtjuk a lekérdezést.
$f_eredmeny = @mysql_query($f_lekerdezes, $kapcsolat)
or die("Lekérdezés hiba! (f_eredmeny) ".gettype($kategoria).mysql_error());
$f_sorok_szama = mysql_num_rows($f_eredmeny);
```

```
//Véletlenszerűen ötöt vagy tizet választunk, figyelembevéve, hogy az adott
kategóriából hány sort eredményezett a lekérdezés..
$szamok[1] = rand(1, $f_sorok_szama);
for($i = 2; $i <= $feladatszam; $i++){
      $van = true;
      while($van) {
            $velszam = rand(1, $f_sorok_szama);
            if(!benne_van($velszam, $szamok)) {
                  $szamok[$i] = $velszam;
                  $van = false;
                  }
      }
}
//Rendezzük a számokat.
sort($szamok);
//Ki kell választani a véletlen számokhoz tartozó sorokat.
switch($kategoria) {
      case 1: {$kat_leiras= "Adat - Információ"; break;}
      case 2: {$kat_leiras= "Adatbáziskezelés - ABKR"; break;}
      case 3: {$kat_leiras= "Rendszertől a Modellekig"; break;}
      case 4: {$kat_leiras= "Egyed - Kapcsolat medell"; break;}
      case 5: {$kat_leiras= "Relációs modell"; break;}
      case 6: {$kat_leiras= "EK átalakítása - Normalizálás"; break;}
      case 7: {$kat_leiras= "Relációs algebra"; break;}
      case 8: {$kat_leiras= "Az Access ABKR alapjai"; break;}
      case 9: {$kat_leiras= "Az SQL nyelv alapjai"; break;}
      case 10: {$kat_leiras= "Vegyes tesztkérdések"; break;}
}
//Rejtett mezőként a feladatszám kiírása.
print "<input type=\"hidden\" name=\"f_feladatszam\"
value=\"".$feladatszam."\"/>";
print "<p id=\"ab_teszt_katkiir\">A választott kategória:
<u>$kat_leiras</u></p>";
//A véletlen számokat tartalmazó tömb alapján a feladatok kiválasztása.
$feladat_sorszam = 1;
foreach($szamok as $szam) {
      $vfazon = mysql_result($f_eredmeny, $szam-1, 'fazon');
      $v_lekerdezes = "select vnev, vleiras from valaszok
      where fazon = '$vfazon'";
      $v_eredmeny = @mysql_query($v_lekerdezes, $kapcsolat)
      or die("Lekérdezési hiba! (v_eredmeny) ".mysql_error());
      print "<div class=\"ab_teszt_feladat\">";
      print "<p class=\"ab_teszt_feladat_p\">$feladat_sorszam.
feladat:</p>";
      print "<p class=\"ab_teszt_feladat_p1\">".mysql_result($f_eredmeny,
$szam-1, 'fleiras')."</p>";
```

```
      print "<input type=\"hidden\" name=\"f_".$feladat_sorszam."\"
id=\"f_".$feladat_sorszam."\" value=\"".mysql_result($f_eredmeny, $szam-1,
'helyes')."\"/>";
//Válaszok kiiratása rádiógombokkal.
      while($v_sor = mysql_fetch_array($v_eredmeny)){
            print "<label><input type=\"radio\"
name=\"RadioGroup_v".$feladat_sorszam."\" id=\"rg_v".$feladat_sorszam."\"
value=\"".$v_sor['vnev']."\">".$v_sor['vleiras']." </label><br>";
      }
      print "</div>";
      $feladat_sorszam++;
      }
}
else {
      print "<p>Kategóriát kell választani!</p>";
} ?>
```

2. számú melléklet: A kiértékelést végző JavaScript programkód

```
//A kitoltott függvény ellenőrzi, hogy minden feladatnál kiválasztottak-e
megoldást.
function kitoltott() {
      //Ellenőrizni kell a JS meglétét!
      if (!document.getElementById) {alert("Engedélyezni kell a JavaScript-
et!"); return false;}
      var feladatszam= document.ab_form.f_feladatszam.value;
      var kitoltve= new Array(feladatszam);
      for(var i= 1; i<= feladatszam; i++) {
            var kitoltott= false;
            var j= 0;
            var db=document.ab_form.elements["RadioGroup_v"+i].length;
            while ((j< db) && !kitoltott) {
                  if
(document.ab_form.elements["RadioGroup_v"+i][j].checked) { kitoltott=
true;}
                  j++;}
            if (j <= db) { kitoltve[i-1]= kitoltott;}
      }
      //Ha van nem kitöltött, akkor figyelmeztetés.
      var i= 0;
      var hianyos= false;
      while ((i< feladatszam) && !hianyos) {
            if (!kitoltve[i]) { hianyos= true; }
            i++;
      }
      if (hianyos) {
            var feladatszamok= "";
            for(i= 0; i< feladatszam; i++) {
                  if (!kitoltve[i]) {
                        var fs= i+1;
                        feladatszamok += fs+", ";
                  }
            }
            alert("A "+feladatszamok+" feladatok nincsenek kitöltve!");
            return false;
      }
      return true;
}
```

```
//A kiértékelő függvény megírása.
function ertekel() {
      if (kitoltott()) {
            var maxpont= document.ab_form.f_feladatszam.value;
            var elertpont= 0;
            kiir_helyes= new Array(document.ab_form.f_feladatszam.value);
            //Nulláról indítom a ciklust a feladatok számozása miatt.
            for(var i=1; i<= document.ab_form.f_feladatszam.value; i++) {
                  //Az űrlapon a feladatsorszám 1-ről indul.
                  var van= false;
                  var j= -1;
                  var
db=document.ab_form.elements["RadioGroup_v"+i].length;
                  while ((j< db) && !van) {
                        j++;
                        if
(document.ab_form.elements["RadioGroup_v"+i][j].checked) { van= true;}
                  }
                  //A j.-et választották ki.
                  if (van) {
                        var valasz= "";
                        switch (j) {
                              case 0: {valasz= "a"; break;}
                              case 1: {valasz= "b"; break;}
                              case 2: {valasz= "c"; break;}
                              case 3: {valasz= "d"; break;}
                        }
                        var jovalasz=
document.ab_form.elements["f_"+i].value;
                        //i-1 a 0-tól induló tömbindex miatt.
                        kiir_helyes[i-1]= i+". feladat: "+jovalasz;
                        if(valasz == jovalasz) { elertpont++; }
                  }
            }
            //Eredmények számítása. A százalék egy tizedesre kerekítve.
            var elertszazalek= Math.round(elertpont*1000/maxpont)/10;
            //Ablak létrehozása, és az adatok kiiírása.
            var maxpont= document.ab_form.f_feladatszam.value;
            var y;
            (maxpont == 10) ? (y =370) : (y = 270);
            var x = window.screen.width-400;
            eredmeny_ablak= window.open("", "eredmeny", "width=400,
height="+y+", screenX="+x+", screenY=150, toolbar=0, menubar=0, status=0,
location=0, directories=0");
            eredmeny_ablak.document.open();
            eredmeny_ablak.document.write("<html><head><title>A teszt
eredményei</title>");
```

```
            eredmeny_ablak.document.write("<link
href='../scripts/basic.css' rel='stylesheet' type='text/css'>");
            eredmeny_ablak.document.write("<meta http-equiv='Content-Type'
content='text/html; charset=utf-8'></head><body>");
            eredmeny_ablak.document.write("<div id='eredmeny_div'>");
            eredmeny_ablak.document.write("<p class='p'>A teszt
eredménye!</p>");
            eredmeny_ablak.document.write("<div class='e'>");
            eredmeny_ablak.document.write("Elért pontszám:
"+elertpont+"<br>");
            eredmeny_ablak.document.write("Százalékos eredmény:
"+elertszazalek+" %<br>");
            eredmeny_ablak.document.write("Maximális pontszám:
"+maxpont+"<br>");
            eredmeny_ablak.document.write("</div>");
            eredmeny_ablak.document.write("<p class='p'>Feladatonként a
helyes eredmények:</p>");
            eredmeny_ablak.document.write("<div class='e'>");
            for(var i= 0; i < kiir_helyes.length; i++) {
                  eredmeny_ablak.document.write(kiir_helyes[i]+"<br>");
            }
            eredmeny_ablak.document.write("</div>");
            eredmeny_ablak.document.write("<form NAME='e_form'><p><input
TYPE='button' VALUE='Ablak bezárása' class='ab_teszt_gomb'
onClick='self.close();'></p></form>");
            eredmeny_ablak.document.write("</div></body></html>");
            eredmeny_ablak.document.close();
      }
}
```

15. Köszönetnyilvánítás

Szeretnék köszönetet mondani tanáraimnak, hogy segítségükkel bővíthettem ismereteimet informatikából.

Különösen szeretném megköszönni Dr. Papp Zoltán Tanár Úrnak az informatika tanárok felé nyújtott három éves munkáját, akitől nem tudtunk lehetetlent kérni.

Szintén külön szeretném megköszönni az órákon, és a szakdolgozat vezetésében nyújtott hasznos segítségét Dr. Adamkó Attila Tanár Úrnak.