# SZAKDOLGOZAT

Czuper László

Debrecen
2008.

**Debreceni Egyetem**
**Informatika kar**

# WEBES ALKALMAZÁSFEJLESZTÉS TERVEZÉSE EGY SZABADON VÁLASZTOTT TÉMAKÖRBEN

Témavezető:

Dr. Kuki Attila
egyetemi adjunktus

Készítette:

Czuper László
mérnök informatikus

Debrecen
2008.

# TARTALOMJEGYZÉK

BEVEZETÉS ..... 4

ALKALMAZOTT TECHNOLÓGIÁK ÉS ESZKÖZÖK ISMERTETÉSE ..... 6
**Az Apache Webszerver** ..... 6
**A MySQL adatbázis-kezelő rendszer** ..... 7
**A PHP** ..... 8

WEBALKALMAZÁSOK TERVEZÉSI ÉS FEJLESZTÉSI LÉPÉSEI ..... 10
**Vízesés modell – fázismodell** ..... 11
**Evolúciós szoftverfejlesztési modell** ..... 12
**Újrafelhasználás-orientált szoftverfejlesztési modell** ..... 13
**Inkrementális szoftverfejlesztési modell** ..... 14
**Spirális szoftverfejlesztési modell** ..... 16

MVC PROGRAMTERVEZÉSI MINTA ISMERTETÉSE ..... 18
Model ..... 18
View ..... 18
Controller ..... 19

WEBES ALKALMAZÁS ISMERTETÉSE ..... 22
**A konkrét alkalmazás tervezése** ..... 22
**Az adatbázis megvalósítása** ..... 24
Az adatbázisban szereplő táblák ismertetése ..... 25
**A rendszer ismertetése** ..... 28
Bejelentkezés ..... 28
Regisztrációs adatok megadása ..... 29

IRODALOMJEGYZÉK ..... 32

# BEVEZETÉS

Az Internet a ’90-es évektől folyamatosan szélesedő elérhetőségével az élet számos területén egyre több lehetőséget kínál az újabb és újabb alkalmazások révén. Az igények bővülésével és a szolgáltatások szélesítésével a világhálón számos információs rendszer is elérhetővé vált.

A mai igények azonban a kezdeti és hagyományos weblapok által nyújtottaknál sokkal kifinomultabb interakciós lehetőségeket feltételeznek. A mai tudás alapú, interaktív, multimédiás webes alkalmazások többsége egy biztos és erősen differenciált és hierarchizált adatbázisra épül. Mára az informatika fejlődésével elmondható, hogy számos lehetőség adott bármilyen adatbázis alapú webes alkalmazás megalkotására, ami persze nem nélkülözi a megfelelő szakértelmet, sőt egyre inkább a szerteágazó ismereteket, a kombinációs, az innovatív és a kreatív képességek.

Az információkat nagyon sokféle alakban találhatjuk meg a weben: van, aki a klasszikus szöveges információkat keresi, van, aki a fényképeket, vagy a multimédiás anyagokat. Ezen szerteágazott keresett információk sokszor több, különböző weblapokon találhatók meg.

Szakdolgozatom célja egy technikai sporthoz (Forma 1) kapcsolódó webes alkalmazásfejlesztés bemutatása, mely elsősorban a szűk értelembe vett felhasználók folyamatosan jelentkező információigényét hivatott csillapítani. Információim szerint, a világhálón nem található olyan komplex, minden igényt kielégítő ilyen jellegű alkalmazás az említett témakörben, így bátorkodtam egyfajta hiánypótló jelleggel az alapjait lefektetni egy ilyen webes alkalmazásnak, a szakdolgozatom szűkös keretein belül.

A megtalálható webes alkalmazások mögött álló adatbázisok komplexitásának hiánya részben e témához kapcsolódóan számos altémakörből, az adatbázis „bonyolultságából”, az ehhez kapcsolódó adatokból, azok variálhatóságából és az adatok sokszínűségéből adódik. Az ilyen alkalmazások tervezése összetett és bonyolult feladat ezért a fejlesztésben számos csoport van jelen a szoftverfejlesztő, a grafikus, a megrendelő illetve a felhasználók csoportja. Széles felhasználói igények jelentkeznek a rendszerrel szemben, így a fejlesztés kezdetén általában

nem lehet teljesen specifikálni, hogy az alkalmazás életciklusa folyamán pontosan miként fog működni, mert struktúrája és funkcionalitása az idő előrehaladtával alakul csak ki.

Ezen okok miatt ismertettem az alkalmazott eszközöket, a szoftverfejlesztés elméletét, illetve a webes alkalmazások globális szerkezetére érezhetően illeszkedő MVC tervezési mintát[1], és ezen szempontokat figyelembe véve próbáltam egy könnyen bővíthető és hordozható, valamint újrafelhasználható rendszert elkészíteni, amely egy jó kiindulási alapot ad egy ilyen összetett rendszer létrehozásához.

[1] MVC=Model view controller egy szoftvermérnöki munkában használt szerkezeti minta.

# ALKALMAZOTT TECHNOLÓGIÁK ÉS ESZKÖZÖK ISMERTETÉSE

## Az Apache Webszerver

*„Az Apache mindig ott volt, ahol a Világháló szíve dobogott - a kezdetektől, amikor az NCSA kiszolgáló szerény oldalhajtásaként megjelent, egészen napjainkig, amikor csak kapkodjuk a fejünket gazdag lehetőségei láttán. Az idők során az Apache képességei számottevően megnőttek, és a programcsomag igencsak összetetté vált - olyannyira, hogy manapság már-már félelmet keltően tornyosul a kezdő felhasználók fölé."*[2]

Az Apache jelenleg a legnépszerűbb webkiszolgáló az Interneten; a Netcraft elemzése szerint a 2008. novemberi adatok alapján a piac 50,34 % is meghaladja.[3]

Választásom azért esett az Apache webszerverre, mert az ingyenessége mellett más előnyös tulajdonságokkal is rendelkezik.

- *Hordozható:* képes működni Linux, Windows, Mac OS X és sok más operációs rendszeren.
- *Rugalmas:* moduláris, bővíthető szerkezettel rendelkezik, és számos módon beállítható.
- *Nyílt forrású*: az Apache ingyenesen letölthető és használható, a forráskód nyíltsága mellett ráadásul azt is jelenti, hogy saját Apache-változatot is készíthetünk.
- *Virtual Hosting*: egyetlen gépen több domainnel rendelkező webszerver futtatása

Az Apache elsősorban statikus és dinamikus weblapok kiszolgálására használható, de számos más feladatban is segíthet, ahol adatokat kell biztonságos és megbízható módon nyilvánosan elérhetővé tenni. Az Apache számos képességgel rendelkezik, közülük sok, mint lefordított modul áll rendelkezésre, amelyek a mag funkcionalitását növelik.

[2] Daniel Lopez, Jesus Bianco: Apache zsebkönyv
[3] http://news.netcraft.com/

## A MySQL adatbázis-kezelő rendszer

A MySQL-t – a világ legnépszerűbb nyílt forrású SQL adatbázis-kezelő rendszerét – a MySQL AB[4] fejleszti, terjeszti és támogatja. A MySQL-ben szereplő SQL a Structured Query

Language, azaz a Strukturált Lekérdező Nyelv rövidítése. A MySQL egy relációs adatbázis kezelő rendszer, az adatok tárolása az adatbázisok tábláiban kapnak helyet, és egy adatbázison belül relációkat definiálhatunk a táblák között.

A MySQl az alábbi előnyös tulajdonságokkal rendelkezik:

- *több tárológéppel* (storage engine) *rendelkezik*: a tárológépeknek megfelelően, a szerver többféle típusú táblát kezel. A fejlesztő a számára a legalkalmasabb táblatípust alkalmazhatja.
- *nyílt forráskódú***:** bárki letöltheti, és ha úgy kívánja, akár módosíthatja is. A MySQL AB a GPL[5] licence szerint szabályozza az adatbázis rendszer használatát.
- *használata nem költséges:* nem kereskedelmi használatra teljesen ingyenes.
- *differencált szerver/kliens architektúrával rendelkezik*: A szerver egy többszálú (multi threaded) SQL szerver, mely sokféle hátteret biztosít (backend), kliensalkalmazást, könyvtárat, alkalmazásprogramozói felületet (application programming iterface, API) és segédeszközt támogat. A szerver és a kliens nem kell, hogy azonos operációs rendszer alatt fusson.
- *kompatibilitása (compatibility) széles:* sok kliens alkalmazás egyszerre teremthet kapcsolatot a szerverrel, a kliensek pedig egyszerre használhatnak több adatbázist.
- *programozhatósága sokrétű*: a szerver sok programozási nyelv részére nyújt programozási interfészt. (pl.: C, C++, C#, Perl, Python, VB. NET, Java, PHP, ASP.NET, stb.)
- *kapcsolhatósága (connectivity) jó*: a szükséges jogosultságokkal rendelkező felhasználó bárhonnan elérheti az adatbázist.
- *hordozhatósága (portability) kiváló*: a szerver sok UNIX dialektus és más operációs rendszer, mint Windows, Mac OS X stb. alatt fut.
- *nagy sebességű használata egyszerű és könnyen elsajátítható*.[6]

[4] AB, a svéd Aktiebolag rövidítése, a magyar RT.-nek felel meg.
[5] GNU General Public Lisence-http://www.fsf.org/licensing/licenses/gpl.html

## A PHP

A PHP[7] nyílt forráskódú, objektum-orientált, gyengén típusos programozási nyelv, a dinamikus, interaktív weboldalak létrehozásának egyik legegyszerűbb és leghatékonyabb eszköze. Használatával elenyésző mennyiségű kódolással egyszerű és hatékony szkripteket illeszthetünk a web oldalunkba, amik a legalapvetőbb feladatoktól a legösszetettebb alkalmazásokig gyakorlatilag bármilyen feladat elvégzésére képesek. A PHP azonban egy jelentős ponton eltér az eddig ismert szkript nyelvektől. Szerveroldali szkript nyelv ez is, vagyis a szerveren fut le, azonban a kód maga a HTML-kódba beillesztve található meg.
Legfontosabb tulajdonsága a nyelvnek az adatbázisok széles körű támogatása.

A PHP-t alapvetően úgy tervezték, hogy rugalmas alkalmas legyen, mivel számos operációs rendszeren használható, együttműködve különböző kiszolgálókkal és adatbázis kezelőkkel. Fejleszthetünk UNIX rendszerre és áttérhetünk NT alapokra minden probléma nélkül. A PHP alkalmazásokat kipróbálhatjuk Personal Web Serverrel, és később telepíthetjük azokat egy UNIX rendszerre, ahol a PHP-t Apache modulként használjuk. A PHP támogatja a kommunikációt más szolgáltatásokkal is különböző protokollok segítségével úgy, mint IMAP, SNMP, NNTP, POP3, HTTP, stb. is.

A PHP 5-tel kerültek bevezetésre az absztrakt osztályok és módszerek, az objektum-orientált nyelvekhez (mint pl. a C++) hasonló standard konstruktor és destruktor létrehozás, valamint a kivételkezelés. Az alapvető objektum-orientált funkciókat már a PHP 3-as verziójába beépítették, de az objektumok kezelését a PHP 5-ben teljesen újraírták a jobb teljesítmény és több lehetőség érdekében.

A PHP lehetővé teszi, hogy a programozók a nyelv funkcióinak bővítése érdekében C nyelven kiterjesztéseket írjanak, amelyeket azután lefordíthatnak a PHP-val együtt, vagy futásidőben, dinamikusan tölthetnek be.

---

[6] Hatvany Béla Csaba: MySQL.Net: MySQL Server adatbázis-programozás .NET környezetben BBS-Info,
[7] PHP: Hypertext Preprocessor egy szerver oldali HTML-be ágyazott szkript-nyelv.

A dinamikus képek és videók generálása során sok más szkriptnyelvhez hasonlóan a PHP forrásai is olvasható formában tárolódnak, ami ugyan nagy rugalmasságot biztosít, de biztonsági és teljesítménybeli problémákat ad. Ezen problémákat tudják orvosolni egyrészt a kód optimalizálók, amelyek csökkentik a lefordított forrás méretét és olyan változtatásokat végeznek el, amivel csökkenthető a végrehajtási idő, másrészt a gyorsítók, amelyek a központi tárba cache-elik a lefordított PHP szkripteket, így elkerülhető azok állandó fordítása minden alkalommal, amikor a szkript lefut.

## WEBALKALMAZÁSOK TERVEZÉSI ÉS FEJLESZTÉSI LÉPÉSEI

A web alapú információs rendszer (Web Information System – WIS) *„egy olyan információs rendszer, amely a weben keresztül biztosítja az interaktív szolgáltatásait és a komplex adatok elérhetőségét*”.[8] Ezen rendszerek „*szolgáltatásaik révén tipikusan az online információelérést és a napi feladatok ellátását támogatják igen nagyszámú (ezres vagy milliós) felhasználói körnek,…*” [9]

A web alapú információs rendszer fejlesztésénél nagy hangsúlyt kell fektetni a tervezésre ezért célszerű absztrakt modellekre bontani. Ezen rendszerek estében is beszélhetünk a szoftver életciklusáról, mely a rá vonatkozó kezdeti igények felmerülésétől a termék megvalósításán, tesztelésén és használatán túl egészen a megszüntetéséig tart.

A fejlesztés folyamatának négy jellegzetes lépését különíthetjük el: a szoftver specifikációt, a szoftver tervezést és implementációt, a validációt és végül a szoftver evolúcióját.
Mit is jelentenek a fent említett a fogalmak?

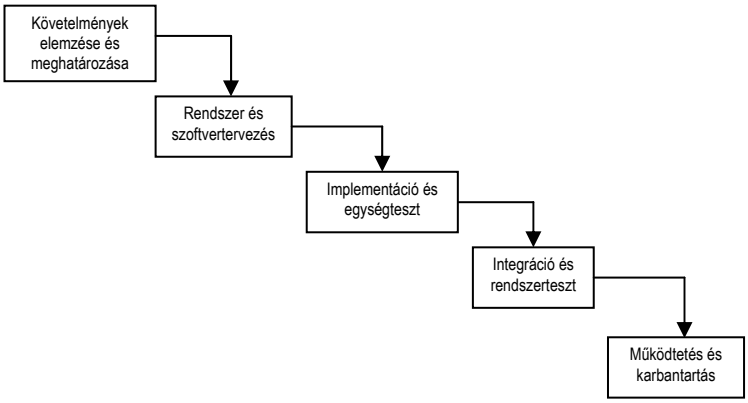
- *Szoftverspecifikáció*: a szoftver funkcióit és annak megszorításait definiálják
- *Szoftvertervezés és implementáció*: a specifikációnak megfelelő szoftver előállítása
- *Szoftvervalidáció*: annak ellenőrzése, hogy azt fejlesztettük ki, amit az ügyfél kívánt
- *Szoftverevolúció*: a szoftver továbbfejlesztése az új igényeknek megfelelően.[10]

A szoftverfejlesztési modellek megértéséhez néhány jellegzetes modellt részletesebben is említek, melyek hasznos útmutatást adhat a szoftverfejlesztés bizonyos részproblémainak megoldása során. Ezen modellek a probléma jellegzetességéből adódóan a rá megfelelő szoftverfejlesztési modell alkalmazható.

[8] Gnaho, C., "Web-based Information Systems Development - A User Centered Engineering Approach." Lecture Notes in Computer Science, 2001., old.: 105-118.
[9] Scharl, A. és Gebauer J. and Bauer, C., "Matching Process Requirements with Information Technology to Access the Efficienty of Web Information Systems." Information Techology and Management 2(2), 2001., old.: 192-210.
[10] „A szoftverfolyamat” Bölecz Mónika előadásvázlata http://www.szt.vein.hu/~bolecz/szf/8.doc

## Vízesés modell – fázismodell

*„A szoftverfejlesztés problémáinak felismerése után a projekt folyamatának leírására kialakult legrégebbi modell a vízesésmodell**,** amelyet fázismodellnek is neveznek. Nevét a szemléltető ábra jellegzetes alakjáról kapta. A modell a termékfejlesztésre koncentrál, azaz az első működő példány előállításáig terjedő szakasz lefolyását ábrázolja. Ma általában a változtatásokat is kezelő ciklikus modellekbe ágyazva használják, önmagában legfeljebb nagy egyedi szoftverek esetén fordul elő.”*[11]




**1. ábra**
Vízesés-modell

Az 1. ábrán bemutatott vízesés-modell lépései a következők:

1. *Követelmények elemzése és meghatározása*: a rendszer szolgáltatásai, megszorításai, céljai a felhasználóval történő konzultáció alapján alakulnak ki. Ezek szolgáltatják a rendszerspecifikációt.

2. *Rendszer- és szoftvertervezés*: A rendszertervezés szakaszában választódnak szét a hardver- és szoftverkövetelmények. Itt kell kialakítani a rendszer átfogó architektúráját.

[11] Kondorosi Károly Szirmay-Kalos László László Zoltán: Objektum orientált szoftverfejlesztés http://www.tankonyvtar.hu/main.php?objectID=5331783

3. *Implementáció és egységteszt*: a szoftverterv programok illetve programegységek halmazaként realizálódik. Az egységteszt azt ellenőrzi, hogy minden programegység megfelel-e a specifikációjának.

4. *Integráció és rendszerteszt:* a különálló programegységek integrálása és teljes rendszerként történő tesztelése

5. *Működtetés és karbantartás:* általában ez a leghosszabb fázis. A rendszer telepítése és használatba vétele után a fellépő hibák kijavítása, a rendszeregységek implementációjának továbbfejlesztése, valamint a szolgáltatások továbbfejlesztése a felmerülő új igényeknek megfelelően.[12]

A vízesés-modell előnye, hogy minden egyes fázisnál részletes dokumentáció készül, ezen dokumentációk révén a rendszer átgondolt, redundáns elemektől mentes, könnyen karbantartható lesz. A modell egyik hátránya az, hogy a követelményeket a rendszer tervezés elején deklarálni kell, és nem feltételezi, hogy ezen követelmények a szoftver életciklusa folyamán változnak, így hibák esetén akár vissza kell menni a legelső szakaszig, kijavítani a problémát.

## Evolúciós szoftverfejlesztési modell

Az evolúciós szoftverfejlesztés alapötlete, hogy először ki kell fejleszteni egy kezdeti implementációt, melyet a megrendelő kipróbál, véleményez, és sorozatos finomítások eredményeképpen végül előáll a végleges rendszer. Az evolúciós szoftverfejlesztést két módon van lehetőség megközelíteni. A feltáró fejlesztés esetében első lépés a követelmények feltárása a megrendelővel közösen, majd a specifikáció egyértelmű részei alapján következik a kész szoftver kialakítása, melyet a megrendelő által – a már működő szoftvert látva, az által inspirálva – kért további funkciókkal kiegészítve elő áll a végleges szoftver. Az eldobható prototípus készítésének célja a megrendelő elképzeléseinek először teljes mértékben történő

[12] „A szoftverfolyamat” Bölecz Mónika előadásvázlata http://www.szt.vein.hu/~bolecz/szf/8.doc

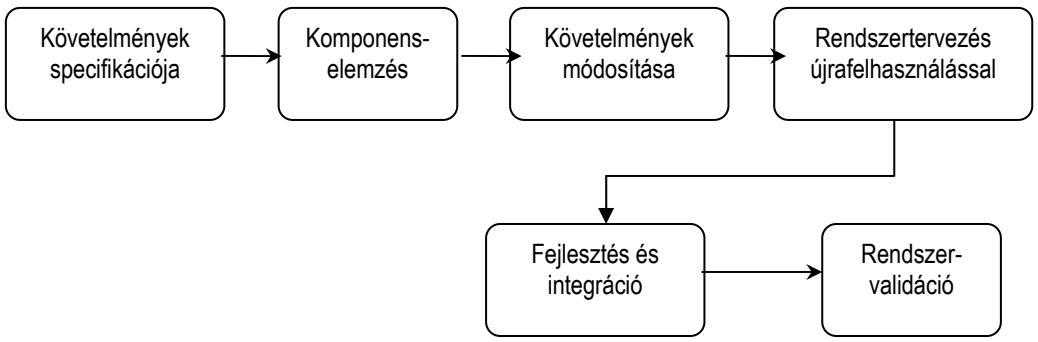
megértése, és azokra alapozva elkészített, mélyebben definiált specifikáció alapján a rendszer kialakítása. Az egyes eldobható prototípusok – próbálgatásos módszerrel – a kevésbé érthető megrendelői követelményekre koncentrálnak, így a prototípusokon keresztül fejlődik a specifikáció, tökéletesedik a követelmények megfogalmazása.

A módszer problémája, hogy az előrehaladás nehezen mérhető, a rendszer a folyamatos módosítások során nehezen átlátható szerkezetű, rosszul strukturált lesz, nem áll rendelkezésre kész specifikáció, és a megfelelő dokumentálás sem biztosított, így a karbantartás nagyon megnehezedik.

A vízesés-modell és az evolúciós szoftverfejlesztési modell azonban egy nagy projektben kombinálható. A specifikáció egyértelmű részeit vízesés-modell szerint, a nem egyértelmű részeket eldobható prototípusokkal, a felhasználói felületet pedig feltáró fejlesztési móddal valósíthatjuk meg.

## Újrafelhasználás-orientált szoftverfejlesztési modell

Az újrafelhasználás-orientált szoftverfejlesztési modellben egy már elkészült implementáció egyes komponensei kerülnek – esetleges átalakítás után – a fejlesztendő rendszerbe való beépítésre. Ilyen módon a fejlesztés jelentősen felgyorsítható. Az újrafelhasználás-orientált modell egyik ismert példája a komponensorientált szoftverfejlesztés, melyben minimalizálható a megvalósítandó szoftverkomponensek száma, de mivel a rendelkezésre álló szoftverkomponensek nem felelnek meg minden elvárásnak, kompromisszumot kell kötni.




**2. ábra**
Komponensorientált szoftverfejlesztés folyamata

A modell lépéseit a 2. ábra szemlélteti, amelynél a komponenselemzés során a követelmény-specifikációban szerepeltek alapján meg kell vizsgálni, hogy mely komponensek implementálták azokat, és a komponensek mely funkcióikban felelnek meg a követelményeknek. Az esetek nagy részében nincs pontos illeszkedés, a felhasznált komponens a funkcióknak csak egy részét nyújtja.

A követelmények módosításánál elemezni kell a követelményeket a komponensek információit felhasználva. Kísérletet kell tenni a követelmények átalakítására az elérhető komponenseknek megfelelően. Ha ez nem sikerül, akkor vissza kell térni a komponenselemzési fázisba és alternatív megoldást keresni.

Rendszertervezés újrafelhasználásával a rendszer szerkezetének kialakítása annak figyelembevételével, hogy milyen komponenseket akarnak újrafelhasználni és együttműködtetni. Fejlesztés és integráció révén a nem megvásárolható komponenseket ki kell fejleszteni és a felhasznált komponensekkel egy rendszerbe kell integrálni.

Előny jelent ennél a modellnél, hogy lecsökkenti a kifejlesztendő részek számát, ezáltal költség, idő és kockázatcsökkentő a hátránya, hogy a követelmények szintjén kompromisszumokat kell kötnünk, így gyakran nem tudunk megfelelni a megrendelő minden kívánságának.

### Inkrementális szoftverfejlesztési modell

A vízesésmodell megköveteli az egyes fázisok véglegesítését a következő fázis elkezdése előtt, amely által egyszerűen menedzselhető, de rugalmatlan módszer. Az evolúciós modell lehetőséget biztosít a döntések elhalasztására, de rosszul strukturált és nehézkesen karbantartható szoftverrendszert eredményez. Az inkrementális szoftverfejlesztés modellje e két módszer előnyeit igyekszik ötvözni, lehetőséget biztosítva az átdolgozások számának minimalizálására, valamint bizonyos döntések későbbre való elhalasztására.

A fejlesztés implementálás része kisméretű egységekben, inkremensekben történik, a mindenkori rendszerbe inkremenseket építünk, inkremensekkel bővítjük, és ha szükséges, nyilván megismételjük az inkremenshez akár a specifikációs tervezés lépését is.

Az alsó rész, az inkremensek hozzáillesztése ciklikus folyamat, mindaddig folytatjuk, amíg úgy nem döntünk, hogy kész van a rendszer. Az inkrementális fejlesztésnek ez a ciklus a lényege. Az inkremenseket önállóan fejlesztjük, tervezzük, implementáljuk, validáljuk, sőt adott esetben még a követelményspecifikációt is megadhatjuk, megváltoztathatjuk inkremensenként. Az architektúrát változatlanul hagyva az inkremensek fejlesztése teljes életciklussal történik.

Az inkrementális szoftverfejlesztési modell előnyei, hogy az inkremenseket úgy kell megtervezni, hogy kis szeletek legyenek, ezekre önmagukban a legmegfelelőbb modell alkalmazható. Továbbá a fejlesztés történhet párhuzamosan is. Kezdhetjük a fejlesztést a inkremensekkel, és mivel az inkremensek kicsik, a felhasználó nagyon hamar kap egy nem eldobható prototípust, amely viszont már nem a követelmény feltáráshoz való, hanem már tudja használni. Tehát bizonyos funkciókat, a legalapvetőbb funkciókat tartalmazó rendszert nagyon hamar kap a felhasználó, a kevésbé lényeges funkciókat majd később beépítjük a rendszerbe. A rendszervalidálás mindig megtörténik, tehát a rendszer ilyen értelemben a beépített inkremensekkel önmagában egy részrendszer, tehát egy használható rendszer.

A rendszerfejlesztés kockázata kisebb, mint az összes többi modellnél, mivel a rendszer validált, kis elemekkel, tehát van egy validált működő rendszer még akkor is, ha befullad a projekt, legfeljebb nem teljes funkcionalitással. Nagyon nagy a valószínűsége, hogy az első pár inkremensnél még nem fogy el a pénz, az idő, az ember, nem változik meg a környezet. Ezért a részsikeres befejezés valószínűbb ennél a fejlesztési modellnél.

A fontosabb funkciókat implementáljuk először, ennek a következtében azokat a funkciókat a felhasználók rendszeresen tesztelik, a fontosabb funkciókat jóval többször használják, mint a kevésbé fontosakat. Így már a rendszerfejlesztés közben a hibák hamarabb kiderülnek, a mindenkori részrendszerben valóban a legfontosabb funkciók a legteszteltebbek. Tehát egy robusztusabb rendszer fejlesztéséhez vezet ez a modell.

## Spirális szoftverfejlesztési modell

Ez teljesen más szoftverfejlesztési modell, mint az eddigiek. Középpontjába olyan dolog kerül, amivel az eddigi modellek nem foglalkoztak: a fejlesztés kockázati tényezőinek felmérése, a kockázatból származó problémákra való felkészülés, a sikertelen kimenetek, és a rizikó csökkentése. Négy alaplépés van, ezeket spirálisan ismételgetjük, tehát a következő spirálban ugyanaz a lépés az eddigieknél magasabb szinten ismétlődik meg.

Az első lépés a követelmények, pontosabban célok meghatározása: az adott spirális fázisban mit fogunk megvalósítani? Azonosítjuk a tevékenységeket, meghatározzuk a megszorításokat, a menedzselési tervet és azt, hogy milyen kockázati tényezők várhatók ebben a lépésben, és ezeket a kockázatokat hogyan fogjuk kezelni, milyen alternatív stratégiák lehetnek a kezelésre.

A második lépés a kockázatelemzés, kockázatok felismerése, kockázatok tényezőinek minimalizálása, vagy annak megszüntetése, ezekre való felkészülés: megtervezzük azokat a lépéseket, amelyek azt célozzák, hogy a kockázatokat el tudjuk kerülni.

Harmadik lépés a fejlesztés és validálás: a második lépés után fejlesztési modellt választunk, adott esetben minden egyes spirálban más-más fejlesztési modell alkalmazható, adott esetben a kockázatanalízis eredményeként választható a fejlesztési modell. Ezek után hajtjuk végre a szokásos tervezés, implementáció, validálás lépéseket. Az első spirálok esetén nyilvánvalóan a követelményfeltárás, követelménytervezés és a validációs lépések következnek, amíg el nem jutunk odáig, hogy áttekintsük a teljes projektet.

A negyedik lépésben az eddigi fejlesztést, az eddigi spirálokat, elemezzük, és döntünk a folytatásról olyan értelemben, hogy befejezzük-e. Amennyiben a folytatásról döntünk, a projektet tervezzük meg, azt tervezzük meg, hogy hogyan folytatódjon a projekt, és kezdődik elölről a spirál.

A spirálmodell előnyei, hogy foglalkozik a kockázatokkal, sokkal szisztematikusabban foglalkozik a projekttel, de hátránya, hogy nem a legtriviálisabb modell, ezért az előnyei nagy rendszereknél hasznosíthatóak.

## MVC PROGRAMTERVEZÉSI MINTA ISMERTETÉSE

A bevezetésben is említettem, hogy törekedtem arra, hogy egy könnyen bővíthető és újrafelhasználható rendszer alapjait fektessem le. Ebben a fejezetben ismertetem az implementáció során figyelembe vett MVC tervezési sablont, mely segítségével egy jól strukturált mintát kaptam, mely lehetővé tette a rendszer komponensekre való bontását, ezáltal biztosítja a későbbi újrafelhasználhatóságot.

Az Model-View-Controllert a Xerox Parc cégnél dolgozták ki a 70-es években. A sablon első, a közvélemény számára is érzékelhető megjelenése a Smalltalk-80 programozási nyelvben volt, a Smalltalk-80 felhasználói felület struktúrájának megtervezéséhez használták.[13]
Az MVC három szóból tevődik össze a dolog, melynek mindnek önálló jelentése van: Model, View, Controller.

### Model

A modellben fogalmazódik meg az üzleti logika. Ezen réteg feladata leírni az adatszerkezeteket, melyeket használni fogunk az alkalmazásban és definiálni azokat a szabályokat, melyek alapján elérhetjük azokat.

### View

A nézet (view) megjelenítéssel/prezentációval kapcsolatos osztályokat írja le, feladata a modell teljes tartalmának vagy egy részének prezentálása. A legfontosabb alapszabály: az adatokat csak a Model osztályainak példányain keresztül érhetjük el és módosíthatjuk**.** Ezt az elvet a helyes tervezés érdekében célszerű követni, akkor az eredményt bármilyen felületen keresztül, legyen az böngésző vagy mobil eszközök alkalmazása, bármilyen tartalmat előállíthatunk a modell alapján.

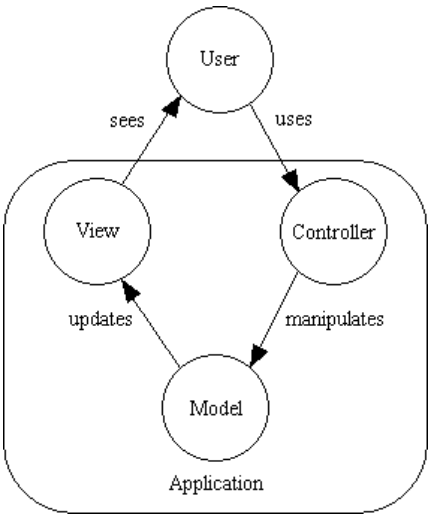
A nézet feladata, hogy fenntartsa a konzisztenciát a modellben tárolt és a nézetben megjelenített adatok között. Ez azt jelenti, hogy amint megváltozik valamilyen adat a modellben, a nézetnek frissíteni kell a megjelenített adatokat. A nézet réteg megvalósítása pedig az alkalmazott technológiáktól függően az alkalmazástól ténylegesen elkülönítve

[13] Programtervezési minták: újrahasznosítható elemek objektumközpontú programokhoz Kiskapu

valósítható meg, sőt, akár több, különböző technikára épülő nézet valósítható meg egyazon alkalmazás számára.

### Controller

A kontroller alakítja át a nézettől érkező felhasználói bemenetet a modell által végrehajtandó parancsokká. A Model és a View-kat össze kellene kötni valahogy, ezért felel a Controller.




**3. ábra**
Általános MVC kapcsolat
(átvéve: http://www.tonymarston.net/php-mysql/model-view-controller.html)

Az MVC előnye, hogy az üzleti logika elválik az interakciós logikától. Az alkalmazás egyes részei jól elhatárolódnak egymástól**,** ezáltal újrafelhasználhatóvá válik a kód, s nem utolsó sorban mindez a csapatmunkát is elősegíti, mindenki a saját feladatára összpontosíthat, a másikra történő túlságosan erős ráutaltság elkerülhető vele. Egyetlen hátránya, hogy jól meg kell tervezni egy nagy adag osztályt, törekedve a bonyolult struktúrák elkerülésére.

A rendszer megtervezése és kivitelezése meglehetősen eltérő tudást igényel, attól, ami a rendszer üzleti logikájának megvalósításához kell, mely ismeretek ritkán vannak meg egy emberben. A rendszer elemeinek szétválasztásával a fejlesztési idő lerövidülhet, ugyanis az egyes részeket a leghozzáértőbb fejlesztők implementálhatják. Az MVC hármas egyes pillérei kellően függetlenek egymástól, így a tervezés után a fejlesztők egymással párhuzamosan dolgozhatnak. Ezáltal lerövidül a fejlesztési idő. A fejlesztés befejeztével a már elkészült

rendszeren az esetleges javítások és módosítások kivitelezése is egyszerűbbé válik. A javítás esetleges mellékhatásai is csak az adott pillért érinti. A hiba okának lokalizálása is egyszerűsödik a rendszer logikus felépítésének köszönhetően.

Az egyes rétegek a lehető legnagyobb mértékben függetlenek egymástól, csak az egymás közötti kommunikáció van specifikálva. Ez a megközelítés elősegíti az egyes rétegek lecserélésének könnyű kivitelezését. Ha például egy, az MVC tervezési sablont támogató rendszert egy másik adatbázis-kezelőre akarunk lecserélni, akkor ezt anélkül megtehetjük, hogy a másik két komponensen bármit is módosítani kellene.

A rendszer hátrányai között meg kell említeni, ha a rendszert alkotó komponensek nagymértékben függnek egymástól, és ha a tesztelés során hibát észlelünk, akkor nem lehet egykönnyen megállapítani, hogy hol található a hiba. Ehhez először meg kell vizsgálni az egyes komponensek közötti kapcsolódásokat, melyeket végigkövetve eljuthatunk a tényleges hibához.

Az egymástól jól elkülönített komponensek esetében teljesen más a helyzet. A tesztelést nem globálisan, az egész rendszerre nézve kell elvégezni, hanem az egyes komponenseket kell validálni.

A tervezett webes alkalmazásomnál (Formula 1 Database), melynél a MVC tervezési sablon szerint készítettem el az implementáció során. Mivel a rendszer felhasználói két csoportot alkotnak – adminisztrátor és felhasználók –, ahol az adminisztrátor módosíthatja a rendszer felhasználóinak adatait, illetve a Forma 1-es adatokkal kapcsolatos karbantartást (bővítés, törlés, módosítás, stb.).

A rendszer felhasználói számára a regisztráció után, elérhető az adatok közötti összetett keresési lehetőségek biztosítása a motorsport világával kapcsolatos interaktív adathalmazokban. Az MVC sablon segítségével a felhasználói és az adminisztrátori felület a rendszer magjának újraírása nélkül a változás könnyen megvalósítható.

Mivel az egyes pillérek teljesen függetlenek egymástól, ezért a forráskód egész nagy részeit újra fel lehet használni. Például a felhasználó jogait ellenőrző rész, vagy közös, adminisztratív funkciók, mint például a felhasználó-menedzsment, személy-kezelés, teljes mértékben átvehető, hiszen az a legtöbb rendszerben ugyanazt a funkciót látja el, a különbségek pedig minimálisak.

# WEBES ALKALMAZÁS ISMERTETÉSE

## A konkrét alkalmazás tervezése

Jelen szakdolgozatban elkészített alkalmazás tervezésekor 3.4 fejezetben tárgyalt inkrementációs szoftverfejlesztési modell előnyeit és hátrányait mérlegelve vettem alapul, a megvalósításkor pedig az előző fejezetben ismertetett MVC sablon alkalmazása mellett döntöttem.

Az inkrementális szoftverfejlesztési modell szerint, így ezen webes alkalmazás szoftverfolyamatának legelső lépése a felhasználói követelmények meghatározása, melynek során a rendszer funkcionalitása kerül nagy vonalakban leírásra. A felhasználói igények és elvárások alapos átgondolása, megfontolása után kialakításra kerül a rendszertől elvárt, a kész rendszer által megvalósítandó funkcionalitás specifikációja.

A tervezett rendszer funkcionalitását tekintve törekedtem arra, hogy olyan felhasználói felületet hozzak létre, melyben a felhasználók az igényeiket jelezhessék, és ezáltal egy komplex későbbiekben minden felhasználó igényeit kielégítő alkalmazássá fejlődjön.

Bizonyos tervezett funkciók csak a tesztüzembe helyezés alkalmával implementálhatók a rendszerbe. Például a Formula 1 versenypályákhoz szorosan kapcsolódó Google Mashup szolgáltatás implementálása. Ez a Google által fejlesztett ingyenes online térképszolgáltatás. Az elérhető térképek és műholdfelvételek az egész Földet lefedik.

A tervezés legelső lépése a felhasználói igények felmérése, azaz annak pontos rögzítése, dokumentálása, hogy a felhasználók milyen igényekkel rendelkeznek, milyen funkcionalitást vár el az elkészítendő rendszertől. Jelen esetben ennek felmérése azáltal valósul meg hogy a rendszer tesztüzembe helyezésével mérni tudjuk a felhasználók ingyenes regisztrálása révén az alkalmazás iránti érdeklődést, illetve az oldal fórumán lehetőség nyílik arra, hogy a felhasználok jelezhessék igényeiket, és ezáltal befolyásolhatják a fejlesztendő alkalmazás funkcionalitását.

Az adatbázis megtervezési folyamatánál több lépést vettem figyelembe. Az egyik fő lépés az optimális adatábrázolás megközelítése, mely leképezendő rendszer elemzésével modellezésével kezdődik. Az adatmodellezés során az adatbázisban tárolni kívánt adatok halmazának ismeretében egy bonyolult adatszerkezet került kialakításra.

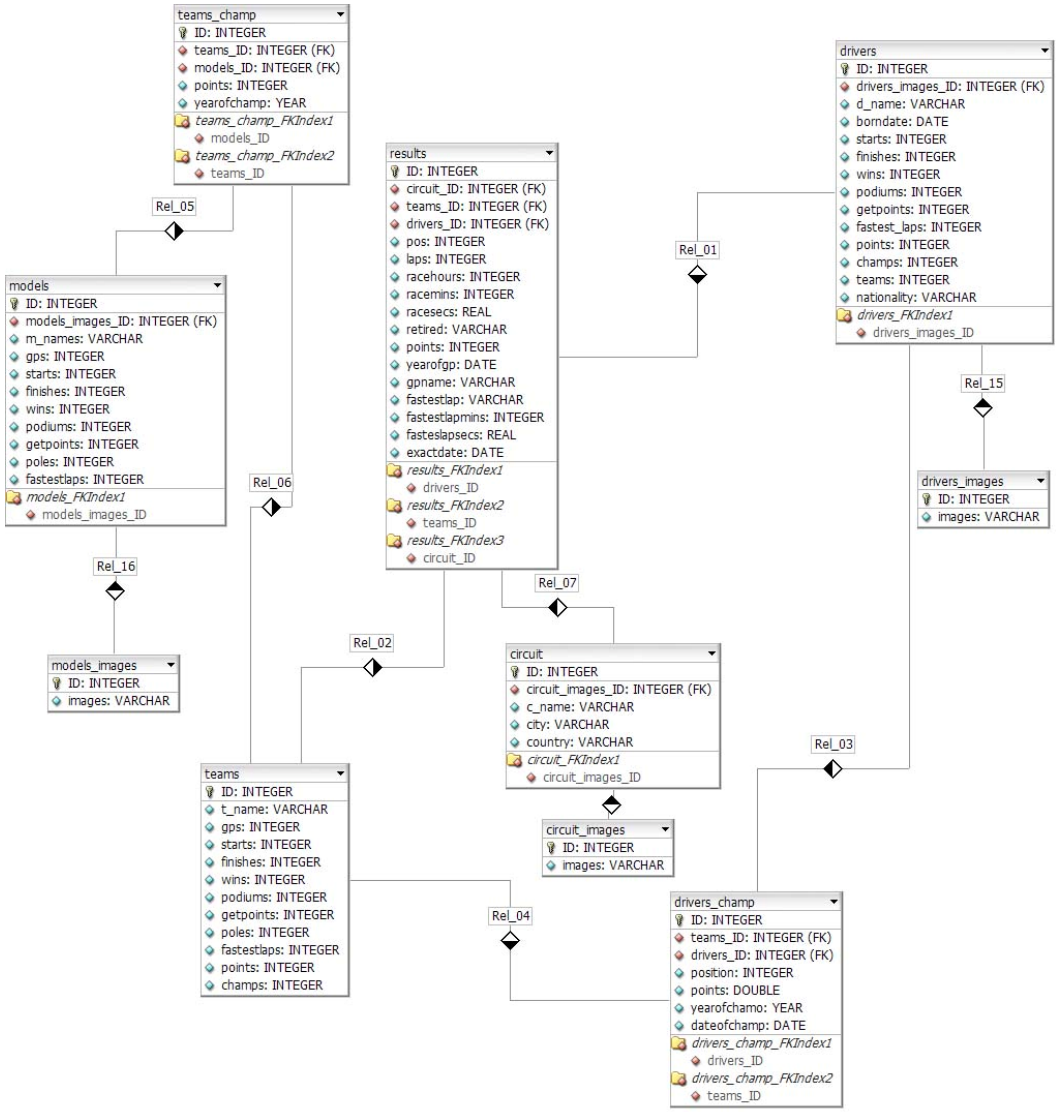
Az adatszerkezettel szemben támasztott követelmény, hogy a táblák létrehozása során kerülni kell a redundáns elemeket, mivel nagyban befolyásolják rendszerünk későbbi működését a szükségtelen tároló terület lefoglalása mellett, mely komplikált adatbázis frissítési és karbantartási műveletekhez vezet, ezáltal könnyen az adatbázis inkonzisztenciáját okozhatják.

A fenti szempont figyelembe vétele mellett az adatbázis-tervezés első lépése a szükséges táblák meghatározása volt. A logikai szint megalkotásakor már figyelembe vettem olyan szempontokat, amelyek már a konkrét használattal kapcsolatosak. A tárolni kívánt információhalmazt táblákra bontottam, majd normalizáltam. A táblák tartalmát úgy terveztem meg, hogy a mezők típusainak meghatározásakor optimalizáltam a táblák közötti kapcsolatokat.

A dinamikus weboldalak elterjedésével, gyakorlatilag a böngészőnkön megjelenő teljes oldal, a megjelenő képek, szövegek, animációk, de még maga a kód is, adatbázisokban tárolható. A képek, vagy a flash videók esetében az adatbázisban csak a fájlok elérési útvonalát tárolom.

## Az adatbázis megvalósítása

A Formula 1 Database rendszer egyetlen központi adatbázist tartalmaz a Formula 1-el kapcsolatos adatok lekezelésére és a regisztrált felhasználók adataira is. A rendszerhez kapcsolódóan az alábbi adatbázist alakítottam ki. A táblák neve többes számú angol főnév lett, a mezőneveket szintén angolul neveztem el. A táblák elsődleges kulcsát minden táblában ’ID’ -val jelöltem. A kacsolódó tábláknál a külső kulcs a ’tablaneve_id’ jelölést követi. A tervezés folyamán a következő táblákat alakítottam ki, melyek között fennálló kapcsolatot a 4. ábra reprezentálja.




**4. ábra**

Az adatbázis-kapcsolatokat realizáló tábla

**Az adatbázisban szereplő táblák ismertetése**

A **'users'** táblában a felhasználók személyes adatai, találhatóak. A felhasználó a következő tulajdonságaikat tudják tárolni: keresztnév; vezetéknév; e-mail; felhasználói név és jelszava, mely a rendszer belépéséhez szolgál; illetve a tábla tartalmaz **status** mezőt, ami lekezeli a felhasználó jogosultságait.

**A 'drivers' (pilóták):**

- **d_name**: versenyző neve
- **borndate:** versenyző születési dátuma
- **starts:** a versenyen indulásainak a száma
- **finishes:** a versenyző befejezett versenyinek a száma
- **wins:** a versenyző győzelmeinek a száma
- **podiums:** a versenyző dobogós helyezéseinek a száma
- **getpoints:** a versenyző pontszerzésinek a száma
- **poles:** a versenyző pole pozicioinak a száma
- **fastest_laps:** a versenyző leggyorsabb köreinek a száma
- **points:** a versenyző összpontszáma
- **champs:** a versenyző világbajnoki cimeinek a száma
- **teams:** a versenyző csapatainak a száma
- **nationality:** a versenyző nemzetisége

A 'drivers_images_ID' külső kulcs segítségével kapcsolódik a 'drivers_images' táblához, amely a pilótákkal kapcsolatos képek statikus útvonalait tárolja.

**A 'models' (konstrukciók):**

- **m_name:** a verseny autók neve
- **gps:** versenyhétvégék száma
- **starts:** indulásainak a száma

- **finishes:** befejezett versenyeik száma
- **wins:** győzelmeik száma
- **podiums:** dobogós helyezéseinek a száma
- **getpoints:** pontszerzéseinek a száma
- **poles:** pole pozicióinka a száma
- **fastest_laps:** gyorskörök száma
- **points:** összpontszám

Mint a ’drivers’ táblánál, itt is a külső kulcs realizálja a kapcsolatot a ’models_images’ táblával, amely a modellekkel kapcsolatos képek tárolására alkalmas.

A **’teams’ (csapatok):**

- **t_name:** a csapat neve
- **gps:** a versenyeik száma
- **starts:** a versenyen indulások száma
- **finishes:** befejezett versenyek száma
- **wins:** győzelmek száma
- **podiums:** dobogós helezések száma
- **getpoints:** pontszerző helyezések száma
- **poles:** pole poziciók száma
- **fastestlaps:** a versenyben elért leggyorsabb körök száma
- **points:** a csapat által elért összpontszám
- **champs:** világbajnoki címek száma

**A ’circuits’ (pályák):**

- **c_name:** a versenypálya neve
- **city:** város neve
- **country:** ország neve

Ezen tábla estében is egy külön táblában tartom nyilván a pályához tartozó pályarajzok

Megjelenítésére szolgáló statikus útvonalakat.

**A ’results’ (verseny eredmények):**

- **position:** a versenyen elért helyezések
- **laps:** a verseny köreinek a száma
- **racehours:** verseny óra
- **racemins:** verseny perc
- **racesecs:** verseny másodperc
- **retired:** a versenyt be nem fejező piloták kiesésének okai (pl.: baleset, motorhiba stb.),
- **grid:** a verseny rajtsorrendje
- **points:** a versenyen szerzett pontok száma
- **yearofgp:** a verseny éve
- **gpname:** a verseny neve
- **fastestlap:** a versenyen elért leggyorsabb kört teljesítő pilóta neve
- **fastestlapmins:** a leggyorsabb kör perce
- **fastestlapsecs:** a legyorsabb kör másodperce
- **exactdate:** a verseny pontos dátuma

Ezen táblában három külső kulcs szolgáltatja a kapcsolatot a ’drivers’, a ’circuit’, illetve a ’teams’ táblákkal.

**’teams_champ’:**

- **position:** a csapat-világbajnokságon elért helyezések
- **points:** a bajnokságban elért pontszámok
- **yearofchamp:** a bajnokság éve

**’drivers_champ’:**

- **position:** az egyéni világbajnokságon elért helyezések
- **points:** az egyéni világbajnokságban elért pontszámok
- **yearofchamp:** a bajnokság éve
- **dateofchamp:** a bajnoki cím megszerzésének pontos dátuma

## A rendszer ismertetése

### Bejelentkezés

A felhasználót a web böngésző címsorába megfelelő kiszolgáló adatainak megadásával az alábbi oldal fogadja (5. ábra).

A regisztrált felhasználók felhasználói nevükkel és jelszavukkal a belépés gombra kattintva azonosításra kerülnek, a user.php osztály segítségével megtörténik a felhasználók validálása. Ha az adott felhasználó szerepel az adatbázisban, a jogosultságainak (adminisztrátor, felhasználó) megfelelő oldal töltődik be.


**4. ábra**
Bejelentkezés

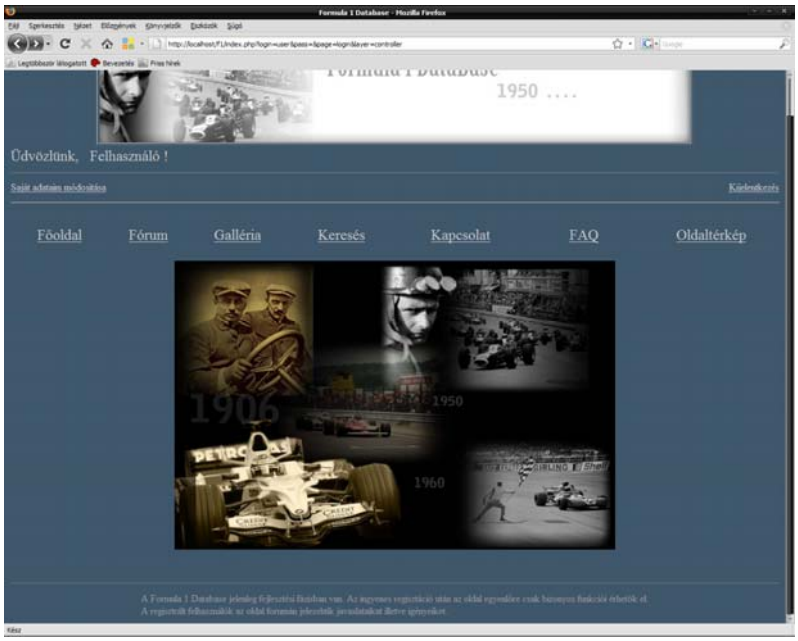
Amennyiben a felhasználó valamilyen hibát vét, akkor a hibának megfelelő üzenet kiírásra kerül.

Regisztrációs adatok megadása

Azon látogatók, akik nem rendelkeznek felhasználói jogosultsággal, a regisztrációs menüpont alatt az alábbi oldalon az adatok helyes megadásával regisztrálhatnak, melyért a signup.php deklarált függvényei felelősek.


**6. ábra**
Regisztráció

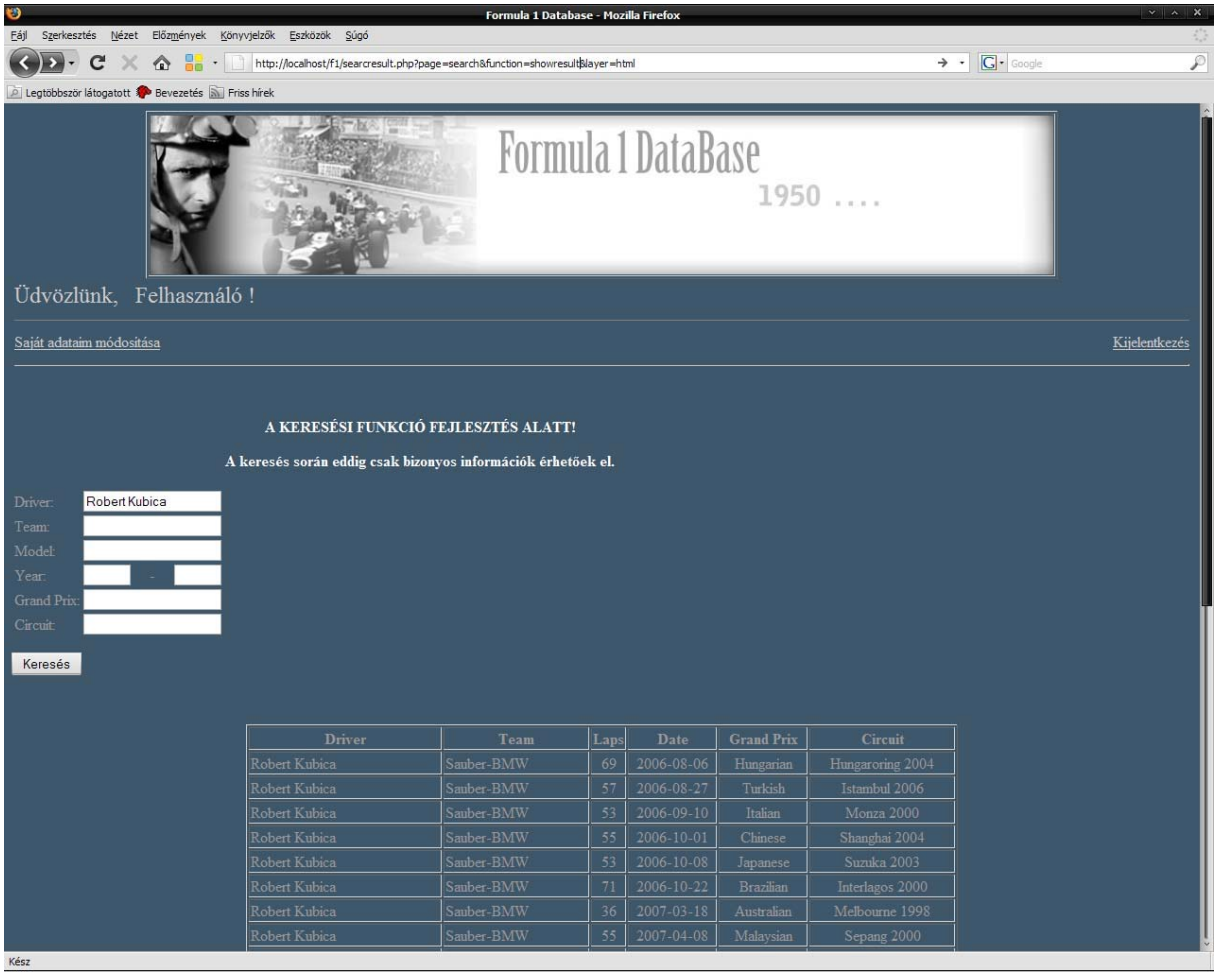
Sikeres regisztráció után a választott felhasználói nevével és jelszóval jelentkezhet be a főoldalra.




**7. ábra**
Felhasználói főoldal

A főoldalon a „saját adataim módosítása” menüpont révén a későbbiekben is meg tudja változtatni a felhasználó az adatait. Kijelentkezés menüpont alatt pedig kijelentkezhet a rendszerből.

A 8. ábra egy részben működő keresési lehetőséget szemléltet.




**8. ábra**
Keresés

# ÖSSZEFOGLALÁS

Szakdolgozatomban nagy hangsúlyt fektetve a webes alkalmazásfejlesztés elméleti hátterének ismertetésére, egy Forma 1 Database nevű alkalmazás alapjait igyekeztem megvalósítani. Ennek oka az volt, hogy szakdolgozatom elkészítése révén betekintést nyertem az általam még nem ismert webes alkalmazásfejlesztés tervezésébe és a létrehozás folyamán felhasznált technológiákba.

Szakdolgozatom többé-kevésbe elérte az általam kitűzött célokat, a dolgozat során igyekeztem jobb betekintést adni a szoftverfejlesztés folyamatába. Ennek elméleti kidolgozása sikeresen megtörtént, a létrehozott alkalmazás folyamatos fejlesztést igényel, tudásomhoz mérten kialakult funkciók, felhasználók lekezelése, illetve a megfelelő felhasználói felületek biztosítása. Mivel ezen webes alkalmazás éles környezetben még nem került tesztelésre, ezt a későbbiek folyamán mindenképpen tervezem a felhasználói igények alapján.

A további funkciókkal való bővítés könnyen megoldható, kivitelezhető, s reményeim szerint a későbbiekben ezáltal egy komplex jól működő rendszert alkot.

Az alkalmazás fejlesztése során rengeteg felmerülő problémával találkoztam, amelyeket részben sikerült orvosolnom, részben további fejlesztést igényelnek. Átnézve az egész alkalmazást, valószínűnek tartom, hogy az adatbázisban tárolt adatok további dekompozícióra, illetve a szoftver későbbi életciklusának előrehaladtával további bővítésre szorulnak.

A témaválasztás során tisztában voltam azzal, hogy egy ilyen rendszer megalkotása összetett és bonyolult folyamat, de számomra nagy kihívást jelentett.

Ezen okok miatt ismertettem az alkalmazott eszközöket, a szoftverfejlesztés elméletét, illetve a webes alkalmazások globális szerkezetére érezhetően illeszkedő MVC tervezési mintát, és ezen szempontokat figyelembe véve próbáltam egy könnyen bővíthető és hordozható, valamint újrafelhasználható rendszert elkészíteni, amely egy jó kiindulási alapot ad egy ilyen összetett rendszer létrehozásához.

# IRODALOMJEGYZÉK


1. Erich Gamma: Programtervezési minták: újrahasznosítható elemek objektumközpontú programokhoz. Kiskapu, 2004.
2. Kondorosi Károly, Szirmay-Kalos László, László Zoltán: Objektum orientált szoftverfejlesztés. Computerbooks, 1999.
3. Jeffrey D. Ullman, Jennifer Widom: Adatbázisrendszerek: alapvetés. Panem, 2008.
4. Hatvany Béla Csaba: MySQL.Net: MySQL Server adatbázis-programozás .NET környezetben. BBS-Info, 2007.
5. Daniel Lopez, Jesus Bianco: Apache zsebkönyv. Kiskapu, 2007.
6. Peter Moulding: PHP haladóknak : fekete könyv. Perfact-Pro Kft., 2002.
7. „A szoftverfolyamat” Bölecz Mónika előadásvázlata
   In: http://www.szt.vein.hu/~bolecz/szf/8.doc Accessed 2008. november 14.
8. http://www.tonymarston.net/php-mysql/model-view-controller.html Accessed 2008. november 7.