SZAKDOLGOZAT

Székely Szilárd

Debrecen
2010

Debreceni Egyetem
Informatikai Kar

Webacar

Témavezető:
Dr. Kuki Attila
egyetemi adjunktus

Készítette:
Székely Szilárd
programtervező informatikus (BSc)

Debrecen
2010

Tartalomjegyzék

1. Bevezető……...5
   1.1 Téma……....5
   1.2 Felhasznált szoftverek, technológiák……..6
   1.3 Szakdolgozat felépítése……...7
2. Rendszerfejlesztés……....8
   2.1 Vízió……....8
   2.2 Követelmény feltárás……...8
   2.3 Elemzés……...9
   2.4 Architekturális tervezés……...9
   2.5 Tervezés……10
   2.6 Implementálás……..……11
   2.7 Tesztelés……....13
   2.8 Üzembe helyezés……...13
3. Technológiák, IDE……..……...14
   3.1 XAMPP……14
      3.1.1 MySQL……14
      3.1.2 PHP ……..14
      3.1.3 Apache ……15
   3.2 JavaScript……..15
   3.3 jQuery……15
   3.4 HTML……...16
   3.5 EditPlus3……...16
   3.6 NaviCat……16
   3.7 dbForge Studio for MySQL……..16
4. Megvalósítás……..17
   4.1 Adatbázis szerkezet……...17
   4.2 Implementálás…..……25
      4.2.1 Sütik ……..…26

4.2.2 Felület, design …….28
4.3 Biztonság …….32
5. Üzleti logika megvalósítása, avagy áttekintés ……...34
5.1 Rendszerkövetelmények ……...35
5.2 Fogalomszótár……...…35
5.3 Szakterületi folyamatok és kapcsolatok ……...…36
6. Program bemutatása képernyőképekkel ……....37
7. Összefoglalás……..43
Irodalomjegyzék…….44
Köszönetnyilvánítás……...……44

1. Bevezető

Jelen világunkat, már el sem lehetne képzelni internet nélkül. Joggal tesszük fel néha a kérdést: Mi lenne, ha holnaptól nem lenne internet? A válaszba egyáltalán nem szeretnék belemenni, igencsak negatív következményei lennének. Manapság szinte minden informatikára épül és az internet adta szabadságra. Ezért is esett a választásom a webes alkalmazásfejlesztés témájára. A másik ok, ami miatt e mellett a téma mellett döntöttem, hogy a webes alkalmazásokkal ki lehet kerülni a platformfüggőség egy részét. Ez alatt a Linuxok, Mac OS-ek és a Windows-ok közötti programfuttatás eltérését értem. Igaz, az egyetemen tanultam Java programozási nyelvről, mely szintén egy ilyen platformfüggetlenség mellett teszi le voksát, de ha egy Java-s alkalmazást szeretnék írni, mely azokkal a funkciókkal rendelkezne, mint amit elképzeltem, akkor a futtatásához, az azt futtató gépnek rendelkeznie kell JVM (Java Virtual Machine)-el. Ez nem olyan mértékben adott még, mint egy böngésző megléte egy operációs rendszeren. Valljuk be, hogy mind a mai napig sok az olyan felhasználó, akik ha meglátnának egy ilyen üzenetet, hogy: „A program futtatásához szükséges a Java Virtual Machine telepítése.” elállnának a program használatától. A webes alkalmazások nagy részének itt a másik előnye: nem kell hozzá más, csak egy böngésző és egy működő internetkapcsolat.

1.1 A téma

A dolgozatomban egy ilyen webes alkalmazást fejlesztek. A címe Webacar. Körülnéztem az interneten és kerestem olyan szoftvereket, melyekből úgymond hiány van a piacon. Nem jártam túlságosan nagy sikerrel, mert ma már szinte mindenre van informatikai megoldás. Egyszer azonban jött egy ötlet a semmiből, melynek utánanéztem és megfelel a kritériumomnak. Ez egy betegszállítói program. Jelenleg Magyarországon csak a MATEJANO Bt. foglalkozik a szoftver fejlesztésével. Mivel ők csak DOS-os alkalmazásként árulják szoftverüket, ezért úgy döntöttem, hogy megpróbálom átültetni a program egy részét a webes alkalmazásba. Azért csak egy részért, mert nagy bonyolultsági fokkal rendelkezik a program. Lehetőségem nyílt konzultálni az egyik betegszállító cég alkalmazottjával, aki

használja ezt a programot és elmondta, hogy a program nagy részét és rengeteg funkcióját nem is használják, de van pár hiányossága. Ezekre próbálok meg odafigyelni, és hogy egy egyszerűen kezelhető, könnyen továbbfejleszthető alkalmazást írjak. Kaptam a cégtől egy betegszállítási adatlapot, valamint egy betegszállítási utalványt. Ezek alapján rögzítik az adatokat a számítógépbe. Úgy gondolom, hogy ez remek kiindulási alap.

1.2 Felhasznált szoftverek, technológiák

A rendszer fejlesztésekor a választott programozói nyelv a PHP lenne. Mára eléggé kiforrott nyelv lett a PHP és tanulmányaim során bár nem találkoztam vele, de önfejlesztés keretében megismertem. Emellett a rendszerfejlesztés technológiája tárgy megadta azt a lehetőséget, hogy egy külsős cégnél végezzem el a tárgy gyakorlatát. A külsős cégnél még jobban elsajátítottam a nyelvet és a benne használatos technológiákat. Használok még továbbá JavaScript-et és jQuery-t. A kód írásakor terveim szerint többnyire EditPlus3 fejlesztői környezetet használok, néha PSPad-ot. A karakterkódolás UTF-8-ban van. Az adatbáziskezelőnek a MySQL-t választottam, az egyszerűség kedvéért. Az adatbázis módosításához a PhpMyAdmin-t vagy a NaviCat-et használom. Szükséges még egy webszerver is, melyet az Apache szolgáltat a 127.0.0.1 címen. A XAMPP nevű programban ezek többsége megtalálható, így nem kell mindet külön-külön feltelepíteni, elég csak ezt az egy programot felrakni és lesz egy működőképes webszerverünk localhost-on. A XAMPP mind Mac OS-re, mind Linux-ra és mind Windows-ra telepíthető könnyedén. Fő teszt operációs rendszerem Windows 7, de egy linux disztribúción, az Ubuntu 9.10-en is tesztelem az alkalmazást. A böngészőnek Firefox-ot használok, de kipróbálás idejére tesztelem az alkalmazást Internet Explorer, Opera és a Chrome böngészőkkel is. Az adatbázistervet, illetve az adatbázis diagrammot a dbForge Studio for MySQL 30 napig ingyenes adatbázistervezővel készítem el. Az UML diagrammokat pedig az ArgoUML szoftverrel állítom elő.

1.3 Szakdolgozat felépítése

Igyekszem nyomon követni a fejlesztés menetét és sorban menni a rendszerfejlesztés lépésein. Dolgozatom első részében bemutatom a technológiákat példákkal együtt, a könnyebb érthetőség kedvéért. Végül képekkel illusztrálva mutatom meg a kész program működését.

2. Rendszerfejlesztés

A rendszerfejlesztést igyekszem a tanultak alapján véghezvinni. Azaz betartani az alább felsorolt fázisok sorrendjét:

- Vízió
- Követelmény feltárás
- Elemzés
- Architekturális tervezés
- Tervezés
- Implementálás
- Tesztelés
- Üzembe helyezés
- Üzemeltetés
- Karbantartás
- Evolúció
- Üzemen kívül helyezés [1]

2.1 Vízió

Ezzel a fázissal kezdem az egész rendszerfejlesztést. Van egy elképzelésem arról, hogy hogy fog kinézni az alkalmazás, miket fog tudni és miket nem. Milyen adatbázist kell felállítanom a háttérben, milyen szintű biztonságot valósítsak meg. Ez csak egy olyan rész, ahol semmilyen kód nem jelenik mag, csak elmélet van a fejemben, esetleg papíron.

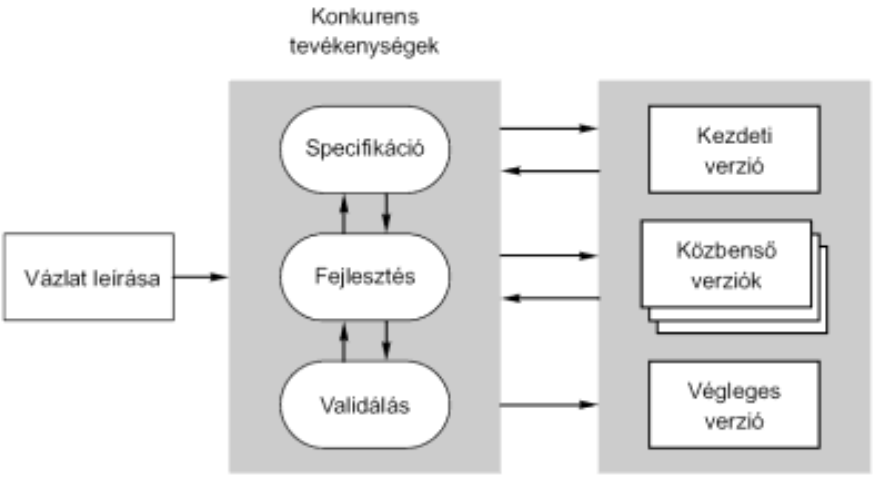
2.2 Követelmény feltárás

A víziómat össze kell vetnem egy hozzáértő emberrel, aki ismer egy betegszállítói szoftvert és kezeli azt. Meglátogatom és megkérdezem arról, hogy hogyan működik a jelenlegi szoftvere, valamint mik az előnyei, hátrányai. Ha lehet, megnézem a felhasználói felületet, hogy az új szoftver ne legyen annyira idegen. Javaslatokat kérek tőle, hogy szerinte

mit kellene változtatni és hogyan. Mik a felesleges dolgok és legfőképp miért használja ezt a szoftvert. Miután ezeket megtudtam, jöhet az elemzés.

2.3 Elemzés

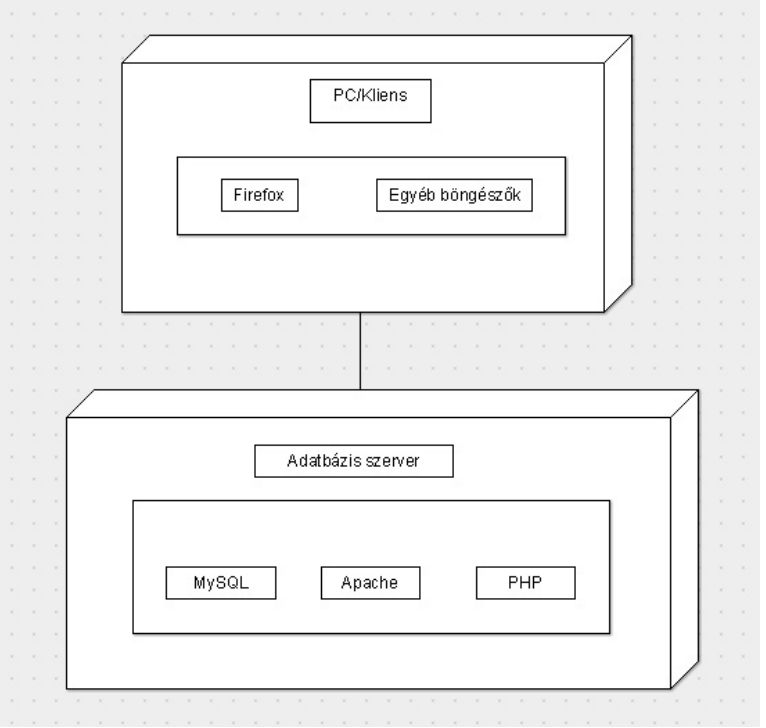
Ebben a részben döntöm el, hogy milyen modell alapján fogom a fejlesztést véghezvinni. Az evolúciós modell mellett döntöttem. Megértettem a követelmények egy részét, ezt specifikálom, implementálom, validálom és kész a prototípus, és ezt addig folytatom, míg az összes követelménnyel nem végeztem. Így lesz egy verziósorozatom, mely egyre több funkciót fed le.[1]



2.4 Architekturális tervezés

Itt átgondolom, hogy melyik adatbáziskezelőt, milyen nyelvet, nyelveket fogok használni, valamint melyik böngészőre építek jobban. Jelen esetben a Firefox böngészőt választom, akkor is, ha az Internet Explorert több ember használja (saját elégedettségből a Firefox iránt és elégedetlenségből az Explorer iránt). Több fejlesztésemben találkoztam már megjelenítési hibával az Explorerben. Itt készül el a Deployment diagramm is. Tisztázom, hogy a kliens/szerver architektúrában mi és hogyan helyezkedik el.

Deployment diagramm:



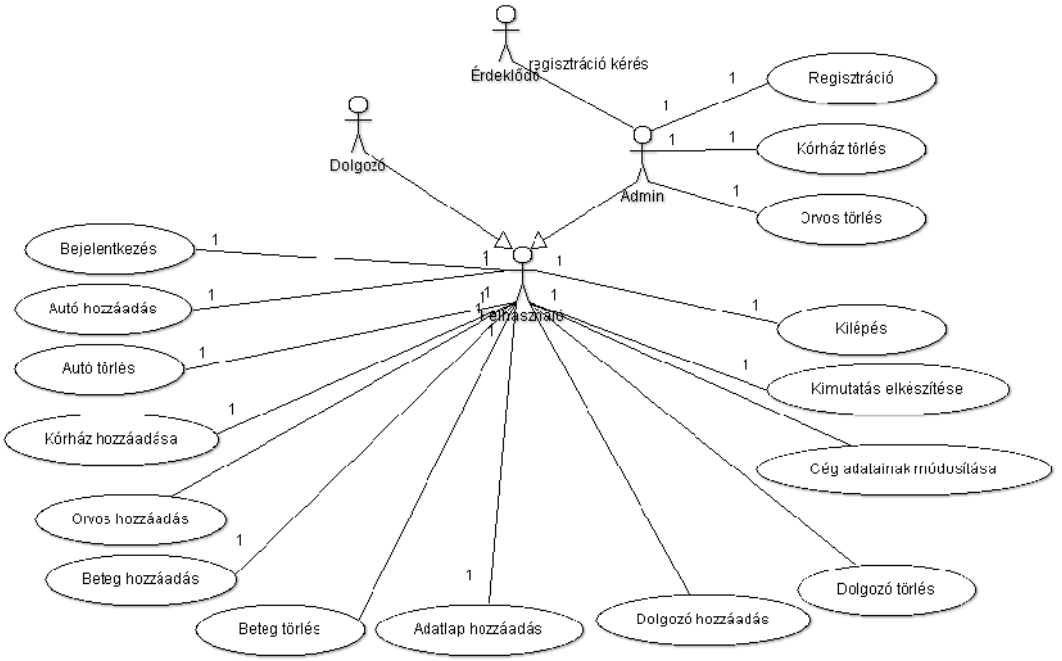
2.5 Tervezés

A felhasználó felület és az adatbázis felépítésének tervezése. A felületnél törekedek egyszerű felhasználói felületre, hogy az átállás az új szoftverre gyors és kényelmes legyen. A színek megválasztásánál figyelek, hogy homogén és szemkímélő színeket használjak.

2.6 Implementálás

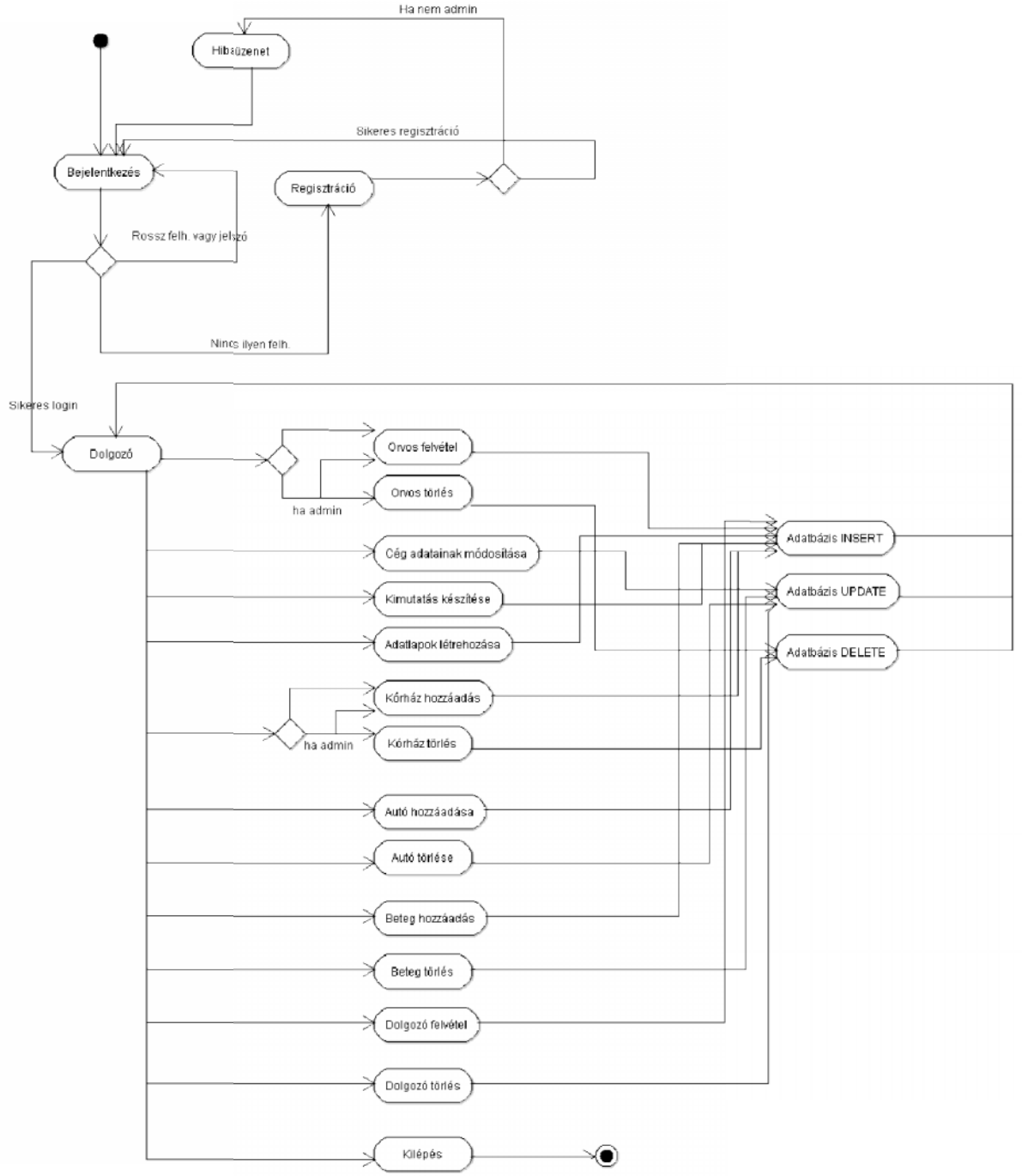
Legelőször feltelepítem a XAMPP-ot a gépemre, így biztosítva Apache-ot, PHP-t és MySQL-t. Elkészítem a kezdeti adatbázist egy táblával és hozzá egy PHP kódot, majd tesztelem, hogy sikeresen tudok-e kapcsolódni az adatbázisom táblájához. Ezután lépésről lépésre hozom létre a táblákat az adatbázisomban, majd ezekhez azokat a php file-okat, amik kezelik az egyes táblákat. A Use-case diagramm előáll és segít átlátni, hogy milyen felhasználók, hogyan használhatják a rendszert, valamint segít az építkezés folyamatában.

Use-case diagramm:



Az activity diagramm pedig arra szolgál, hogy lássuk magát a folyamtatot, hogy felhasználás és a program működése hogyan történik. Ez is nagyban hozzájárul a fejlesztés menetének gyorsításához.

Activity diagramm:



2.7 Tesztelés

Minden elkészült verzió után tesztelem a kódom, hogy minden rendben van-e. Figyelek a biztonságra is, hogy illetéktelen felhasználó ne használhassa a programot.

2.8 Üzembe helyezés

Ha elkészült az RC verzió(Release Candidate – Kiadásra jelölt változat) akkor egy nyilvános webszerverre feltöltöm a file-okat, valamint létrehozom ott is ugyanazt az adatbázist ugyanazzal a táblákkal. Egyedüli dolog, amire figyelnem kell a kapcsolódás az adatbázishoz, ugyanis már nem localhost-hoz kell kapcsolódnom, hanem ahhoz a kiszolgálóhoz, amit nekem megadnak. Természetesen ekkor a jelszó is változik.

A további életciklusokra nem szeretnék kitérni, mert az majd a jövő feladata lesz.

3. Technológiák, IDE

3.1 XAMPP (X – cross-platform, A – Apache HTTP Server, M – MySQL, P – PHP, P – Perl)

Segítségével nem kell külön telepítéseket, beállításokat végeznem a gépemen, hogy működjön egy tesztkörnyezet, melyen gyorsan tudom kipróbálni az általam készített szoftvert.

Jelenleg az 1.7.1-es verziót használom.

3.1.1 MySQL

A világ legnépszerűbb nyílt forráskódú adatbázis kezelője. Többfelhasználós, többszálú SQL alapú relációs adatbázis-kezelő. Költséghatékony és egyszerűen beállítható megoldást ad dinamikus webhelyek szolgáltatására. [2] Bár 2010. január 27-én az ORACLE felvásárolta a SUN-t, mely forgalmazta a MySQL-t 2008 óta, így kérdéses az online körökben, hogy meddig marad ingyenes a szoftver.

Verziószám: 5.0.51a

3.1.2 PHP

A PHP nyílt forráskódú, számítógépes szkriptnyelv, legfőbb felhasználási területe a dinamikus weboldalak készítése. Emiatt a PHP-t jórészt szerver-oldalon használják, bár létezik parancssori interfésze is, illetve önálló, grafikus felületű alkalmazások is létrehozhatók vele. A PHP oldalak elkészítésénél a HTML-t gyakorlatilag csak mint formázást használják, ugyanis ezen lapok teljes funkcionalitása a PHP-re épül. Amikor egy PHP-ben megírt oldalt akarunk elérni, a kiszolgáló először feldolgozza a PHP utasításokat, és csak a kész (HTML) kimenetet küldi el a böngészőnek, így a programkód nem is látható kliens oldalról. Ehhez egy ún. interpretert (értelmezőt) használ, amely általában egy külső modulja a webszervernek. A PHP nyelv lényegében nagymértékű kiegészítése a HTML-nek, ugyanis rengeteg olyan feladat végezhető el vele, amelyre az ügyféloldali szkriptek nem képesek (vagy ha igen,

korlátozottan). Ilyen például a bejelentkezés, az adatbáziskezelés, filekezelés, kódolás, adategyeztetés, kapcsolatok létrehozása, e-mail küldése, adatfeldolgozás, dinamikus listakészítés stb. Minden olyan esetben, ahol nagyszámú ismétlődő feladatsort kell végrehajtani (például képek listázása és linkelése, listakészítés stb.), ott ez a programnyelv nagyszerű segítség.[3]

Verziószám: PHP/5.2.9

3.1.3 Apache

Az Apache HTTP Server (röviden Apache) egy nyílt forráskódú webkiszolgáló alkalmazás, szabad szoftver, mely kulcsfontosságú szerepet játszott a World Wide Web elterjedésében. A projekt célja egy olyan webszerver program létrehozása, karbantartása, és fejlesztése, amely megfelel a gyorsan változó Internet követelményeinek. Biztonságos, üzleti, vállalati felhasználásra is megfelelő és szabadon használható. [4]

Verziószám: Apache/2.2.11

3.2 JavaScript

A JavaScript programozási nyelv egy objektumalapú szkript nyelv, amelyet weblapokon elterjedten használnak. Dinamikus oldalaknál továbbfejlesztett felhasználói felület létrehozásában nyújt segítséget.[5]

3.3 jQuery

A jQuery egy népszerű JavaScript library, mely a HTML kód, és a kliensoldali JavaScript közötti kapcsolatot hangsúlyozza. A jQuery filozófiája, hogy amennyire csak lehetséges leválassza a JavaScript kódot a HTML-ből, és különböző eseményvezérlőkön, és azonosítókon keresztül kommunikáljon a weblap HTML elemeivel.[6]

3.4 HTML

A HTML (HyperText Markup Language) egy leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki, és mára már internetes szabvánnyá vált a W3C (World Wide Web Consortium) támogatásával.[7]

3.5 EditPlus 3

Az EditPlus3 egy szöveg-, HTML-, PHP és Java szerkesztő Windows operációs rendszerre. A különböző programozási nyelvek, különböző szín kombinációkkal kiemelve vannak jelölve, ezzel segítve a könnyebb programozást. Fülek használata, automatikus kódkiegészítés, automatikus formázás teszi még könnyebbé a munkát. Az ok, amiért ezt a szerkesztőt használom, az nem más, mint hogy az ékezetes magyar betűket át lehet konvertálni HTML-nek megfelelő kódolásúra és már hozzá szoktam a használatához régebbről.

3.6 NaviCAT

A NaviCAT programot a MySQL adatbázisom gyors és egyszerű eléréséhez használom, hogy tesztadatokat vigyek be a rendszerbe. Letisztult felhasználói felülettel rendelkezik ezzel is segítve a gyors munkát.

3.7 dbForge Studio for MySQL

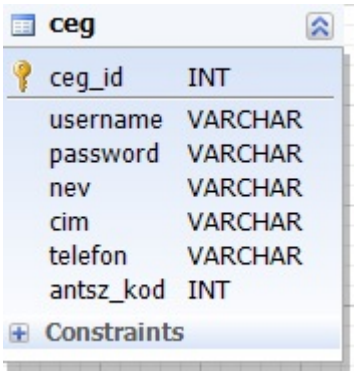
Egy 30 napig ingyenes szoftver, melyet nemrég ismertem meg és szép diagrammokat lehet vele szerkeszteni. Hasonló még a MicroOLAP szoftver is, csak mire eljutottam oda, hogy a kész adatbázisból generáljak egy rajzot, addig lejárt a kipróbálási időm. Így maradt a dbForge Studio for MySQL. Nem csak rajzokat lehet vele összeállítani, hanem módosítani is lehet egy meglévő adatbázison is.

4. Megvalósítás

4.1 Adatbázis szerkezet

Első lépésként tisztázom, hogy kik lesznek a felhasználói, a programomnak. Egy olyan szoftvert szeretnék megvalósítani, amit minden egyes betegszállítói cég használhat. Itt tisztában kell lennem azzal, hogy a programomat adatrögzítésre és kimutatásokra fogják használni. Tehát kell egy olyan tábla, amiben tárolom a szoftvert használó cég adatait. Ebben lesz egy elsődleges kulcs, mely alapján egyértelműen azonosítani lehet majd egy céget. Egy cég kap felhasználónevet és jelszót, amivel be tud jelentkezni a programba és használni tudja azt. A jelszót titkosítva tárolom, hogy még ha illetéktelen kezébe kerül is a jelszó, ne tudja könnyen megfejteni. MD5-ös titkosítást használok, mert ezt a módszert használják a legtöbb helyen. Tárolom még a cég nevét, telefonszámát, címét és az ÁNTSZ kódját, azért, hogy a kimutatásokban a nyomtatáshoz hozzá lehessen fűzni. A megszorítások pedig a következők: a ceg_id INT típusú és 11 hosszúságú. Ez volt az alapértelmezett érték, így ezt nem változtattam. A username VARCHAR típusú és 20 hosszúságú, míg a password mező 32 hosszúságú az md5 miatt. Bár a bemeneti jelszó, amit az ügyfél választ, csak 15 hosszúságú lehet. A név és a cím szintén VARCHAR típusú és 70 hosszúságot engedélyez. A telefon mező is VARCHAR lett, mert előfordulhat olyan, hogy valaki nem csak számokat használ telefonszámának megadásakor. Pl.: +36-30-111-2222. Az ÁNTSZ-kód viszont már INT típusú és 9 hosszúságú.

Lássuk a táblát:

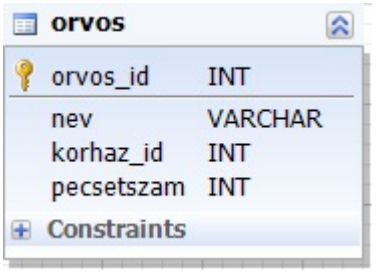


Egy betegszállítói cégnek szüksége lesz két olyan táblára melyet közösen is használhatnak a konkurens cégek. Ez a két tábla a kórház tábla és az orvosok tábla. Erre azért van szükség, mert a betegszállítói lapon fel kell tüntetni, hogy egy beteget melyik egészségügyi helyszínre rendelte be melyik orvos. A kórház táblában tároljuk a korhaz_id-t, ami a tábla elsődleges kulcsa. Tárolásra kerül a kórház neve, osztálya. Külön oszlop lesz még a város, a cím, és az ÁNTSZ-kód. Megszorítások: korhaz_id INT típusú és 11 hosszú. A nev és a varos VARCHAR típusú és 50 hosszú. A cím ugyanúgy VARCHAR típusú, de 80 hosszú. Az antsz_kod itt is 9 hosszú INT.

A tábla:

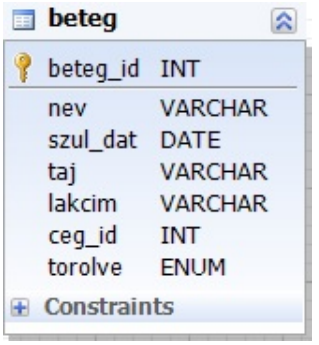
Az orvos tábla annyiban lesz más, hogy megjelenik az első külső kulcs a programban. Az elsődleges kulcs itt is értelemszerűen az orvos_id lesz, az egyértelmű azonosítás miatt. Ebben a táblában nem tárolunk semmi különöset az orvosról, csak a nevét és a pecsétszámát. A fentebb említett külső kulcs a korhaz_id lesz. Erre azért van szükség, hogy az orvosokat kórházakhoz tudjuk kötni. 1:N kapcsolat áll a két tábla között, azaz 1 kórházhoz több orvos tartozhat. Megszorítások: nev 50 hosszú VARCHAR, orvos_id, korhaz_id, pecsetszam 11 hosszú INT.

Tábla:



Következő táblám a beteg tábla lesz, melyben a szállított betegeket tárolom le. Ez a tábla viszont az orvos és a korhaz táblával ellentétben, már nem lesz látható minden felhasználó számára. Csak azon táblarészeket látja a felhasználó, amit az ő cége hozott létre. Ezért itt is megjelenik egy külső kulcs, méghozzá a ceg_id. Ezen külső kulcs segítségével szelektálom majd a lekérdezést. Megtalálható a táblában a beteg neve, születési dátuma, TAJ száma és lakcíme. Még egy mezőt azonban találhatunk a 2 kulcsmezőn kívül is. Ez lesz a torolve mező. Ez egy ENUM típusú mező. Értékei: true és false. Azt mutatja meg, hogy ezt a beteget törölték-e logikailag az adatbázisból vagy sem. Azért valósítok meg logikai törlés, hogy a későbbiek folyamán az adatbázisom konzisztens maradjon, és ne hivatkozzak olyan kulcsú betegre, aki nem létezik. Ez csak egy olyan szolgáltatás, amivel lehet logikai törlést csinálni, nem feltétlenül kell ezt használni a cégeknek. Megszorítások a többi mezőre: nev VARCHAR 50 hosszú. A szul_dat DATE típusú. A taj 9 hosszúságú INT, mely szintén egyértelműen azonosít egy beteget. Későbbiek folyamán ezt ki fogom használni. A lakcim 60 hosszú VARCHAR. és a két kulcs 11 hosszú INT.

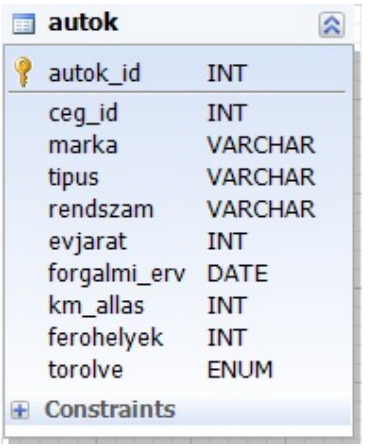
Tábla:



Következő táblám az autok tábla. Itt tároljuk a cég autóit. Ezt szintén úgy kellett megoldanom, hogy a cégek ne láthassák az összes autót, aki betegszállítással foglalkozik, így ide is bekerült a ceg_id külső kulcs. 1 céghez több autó is tartozhat. A táblában megtalálhatóak még a marka, tipus, rendszam, evjarat, forg_erv, km_allas, ferohelyek és a torolve mezők. Ezeket értelem szerint kitölti a felhasználó. a regisztráció után, különben nem fog tudni adatlapokat bevinni a rendszerbe, ha nincs legalább egy betegszállító autója. Érdekességként a km_allas mezőt emelném ki, ugyanis a rendszer ezt a mezőt frissíti a bevitt adatlap szerint. A másik, hogy itt is megtalálható a torolve mező, így ebben a táblában is lehet

logikai törlést véghezvinni. Megszorítások: auto_id, ceg_id 11 hosszú INT. marka: 15 hosszú VARCHAR, tipus: 25 hosszú VARCHAR, rendszam: 10 hosszú VARCHAR, evjarat: 4 hosszú INT, forgalmi_erv: DATE, km_allas: 6 hosszú INT, ferohelyek: 2 hosszú INT.

Tábla:

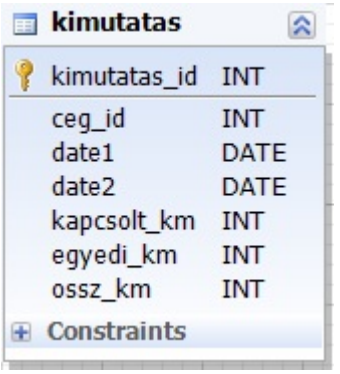


A következő tábla a dolgozó tábla, ahol nyilvántartjuk a cég dolgozóit. Ez egy olyan pontja az alkalmazásnak, amiből több kimutatást lehet készíteni a jövőben. Itt szintén nem szeretné senki, ha mindenki látná az összes dolgozót, aki betegszállítással foglalkozik, így itt is megjelenik a ceg_id külső kulcs, valamint a torolve mező, a logikai törlés miatt. Megszorítások: dolgozo_id, ceg_id 11 hosszú INT, nev: 30 hosszú VARCHAR, fizetes: 7 hosszú INT, telefon: 15 hosszú INT, email: 30 hosszú VARCHAR, adoszam: 9 hosszú INT, szem_szam: 9 hosszú VARCHAR. Jelen táblának egyetlen mező neve rövidített: szem_szam. Itt a dolgozók személyi igazolvány számát tároljuk le.

Tábla:

Az utolsó előtti tábla a kimutatas tábla. Ezt a táblát azért hozom létre, hogy bármilyen kimutatást, amit egy cég létrehoz, az a szoftverben visszakereshető legyen. A táblát az interjún elhangzottak alapján alakítottam ki. Az interjún megkérdeztem, hogy mire használják a betegszállítói szoftvert. A válasz az volt, hogy az összegyűjtött adatok alapján havi zárást kell csinálniuk, hogy az autóik hány kilométert tettek meg egyedi betegszállítással és kapcsolt betegszállítással. Az egyedi betegszállítás azt jelenti, hogy az autóban csak 1 beteg ül és egyedül vele teszik meg az utat. A kapcsolt betegszállítás pedig az, hogy több beteg ül az autóban, akiket különböző helyen vesznek fel, majd így szállítják haza vagy az egészségügyi helyszínekre őket. A táblában már a jól ismert ceg_id szerepel, így kötve egyes kimutatásokat egyes cégekhez. Megszorítások: kimutatas_id, ceg_id 11 hosszú INT, date1, date2 DATE típusú mező, melyben a kimutatás kezdeti és végpontja szerepel. Mondjuk egy havi zárás esetén a date1 mező:2010-02-01 míg a date2 mező:2010-03-01. Így a program megmondja, hogy az adott cég a februári hónapban hány kilométert tett meg egyedi, illetve kapcsolt betegszállítással. A maradék három mező: kapcsolt_km, egyedi_km, ossz_km 11 hosszú INT.

Tábla:



Végül jöjjön a legtöbb mezőt tartalmazó tábla, melyet a betegszállítói adatlap alapján válogatom össze. Úgy alakítom ki a szerkezetet, hogy csak azokat az adatokat tároljam, amik a betegszállítói lapon szerepelnek, de azokból sem mindent, mert az interjú alatt kiderült, hogy azon is vannak felesleges mezők. Azért, hogy ne tároljam ugyanazt az adatot kétszer

sehol, így ebben a táblában több külső kulcs is szerepel. Először is a szokásos ceg_id, mellyel céghez kötöm az adatlapokat. Auto_id, ami azért kell, mert rögzítésre kerül a betegszállító autó és annak kilométer állása. Van két darab dolgozo_id, egy a sofőr és egy az ápoló részére. Van két korhaz_id, amiatt, mert nem biztos, hogy a berendelő intézmény ugyanaz lesz, mint ahová beviszik a beteget. Ugyanezen ok miatt 2 orvos_id is szükséges, mert nem biztos, hogy ugyanaz az orvos veszi át a beteget, mint aki berendelte. A külső kulcsok után jöjjenek a többi mezők. A betegszállítói adatlapon megtalálható egy úgynevezett BNO kód, amely a betegség leírására szolgál. Ez egy 5 jegyű INT lesz. Több ENUM típusú mezőt készítek, hogy a bevitel gyorsabb legyen. Ezek a következők: kisert, ahol megadható, hogy van-e betegkísérő vagy nincs. A szállítási módot szall_mod-nak hívom és értékei: ’ulve’, ’fekve’. Van egy surgos mező, melynek értékei:’6’, ’24’, ’idore’. A hatos azt jelenti, hogy 6 órán belül kell a beteget a célállomásra szállítani, míg a 24 pedig azt jelenti, hogy 24 órán belül és az időre pedig az egyéb kategória, amit az adatlapon feltüntetnek, de az adatbázisban erre nincs szükség. Akad még egy koltsegviselo mező, ahol az értékek: ’beteg’, ’oep’, ’egyeb’. Ez a mező arra szolgál, hogy ki állja a betegszállítás költségét. A legtöbb felsorolt kategóriával a szall_kat mező rendelkezik. Ebben az esetben nem nevesítettem az értékeket, csak 0-tól 8-ig adtam meg számokat. Magát azt, hogy melyik szám mit jelent, a felhasználói felületen lehet látni. De a felsorolás kedvéért álljanak itt rendre értékeik:

- 0 - magyar biztosítás alapján végzett ellátások
- 1 - magyar biztosítással nem rendelkező menekült ellátása
- 2 - államközi szerződés alapján végzett ellátás
- 3 - egyéb, magyar biztosítással nem rendelkező vagy más hatályos rendelkezés alapján magyar egészségügyi ellátásra nem jogosult személyek térítésköteles ellátása
- 4 - magyar biztosítással nem rendelkezők menedékes ellátása
- 5 - külföldön élő magyarok központi költségvetésből támogatott ellátása
- 6 - befogadott külföldi állampolgár
- 7 - menekült, menedékes státuszt kérelmező
- 8 - elszámolásra vonatkozó nemzetközi szerződés alapján történő ellátás

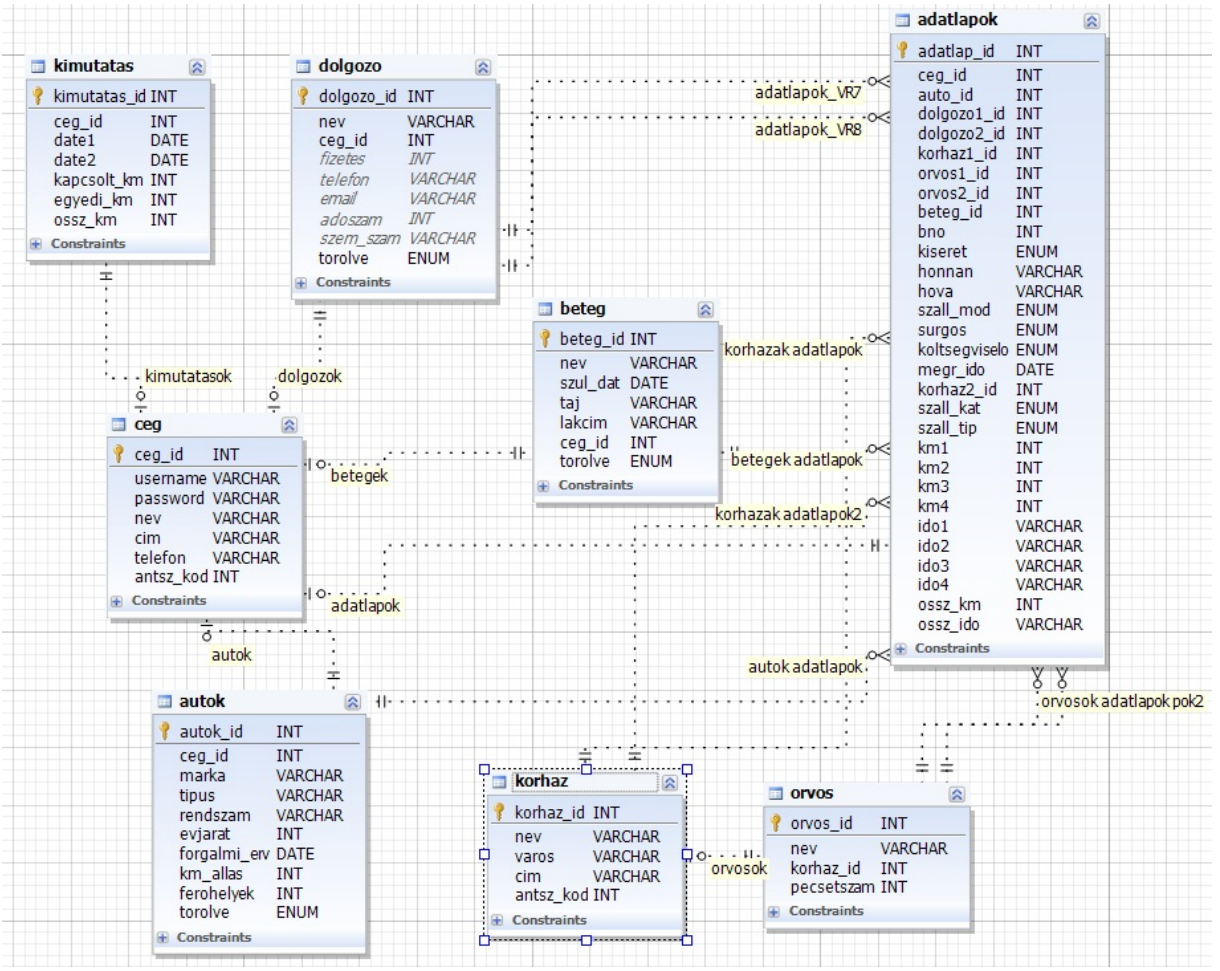
További mezők, még a megr_ido, ami a megrendelési idő rövidítése. Ez is a kimutatások egyik kulcsmezője, ugyanis ez alapján a dátum alapján szelektálunk. Az utolsó ENUM típusú

mező a szall_tipus, aminek értékei az ’egyedi’ és a ’kapcsolt’. Ezek a nevek már ismerősek, ugyanis a kimutatások legfontosabb szelektáló tényezője ez a mező. Kimaradt még 4 db INT 6 hosszúságú mező, melyben a kilométereket tároljuk. Azért kell 4 db belőle, mert minden egyes adatlapon rögzíteni kell a telephelyről való indulás kilométer állását, a beteg felvételekor és az átadásakor is fel kell ezt az értéket írni, valamint mikor visszaérkezik a telephelyre az autó, akkor is. Ugyan ez a helyzet az idővel is. ezt is dokumentálni kell. Ezeket az értékeket 4 db VARCHAR 5 hosszúságú mezőben tároljuk, ugyanazon az elven, mint a km állást. Végül mind az időnek és mind a megtett kilométerek összesített értékét fel kell írni. Ezek számára ugyanazon típust választottam, mint az előzőeknél.

Tábla:

Fentebb említettem pár helyen a táblák kapcsolati fokát, de a következő diagrammon szeretném bemutatni a táblák kapcsolatát. Igyekeztem úgy rendezni a táblákat, hogy a köztük levő nyilak minél jobban látszanak.

Az adatbázis táblák kapcsolata:



4.2 Implementálás

Ebben a részben szeretném bemutatni, hogyan valósítottam meg a betegszállítói szoftvert PHP, HTML, CSS, JavaScript, jQuery és SQL segítségével.

Elsőként az adatbázishoz való kapcsolódást írom meg, egy conn.php file-ban, amit így nem kell mindig újraírni, hanem csak include-olom a többiben.

A conn.php tartalma:

```
<?php
    header ('Content-type: text/html; charset=utf-8');
     mysql_connect("localhost","webacar_ver1","webacar_jelpwd")
     or die(mysql_error());
     mysql_select_db("webacar_ver1") or die(mysql_error());
?>
```

A header azért került ide, hogy ezt se kelljen minden állomány elejére írnom. A header-nek itt az a feladata, hogy közölje a böngészőkkel, hogy az általuk megjeleníteni kívánt lap milyen karakterkódolásban van. Mint dolgozatom elején mondtam, az UTF-8 mellett döntöttem. A mysql_connect-ben a paraméterek a következők: az első paraméter mondja meg a webszerver címét, a második paraméterben a felhasználót választjuk ki, akinek hozzáférési engedélye van az általunk későbbiekben kiválasztandó adatbázishoz, míg a harmadik paraméterben a felhasználóhoz tartozó jelszót adom meg. A mysql_select_db-ben választom ki, hogy melyik adatbázishoz szeretnék kapcsolódni. Mindkét mysql függvényhívás után található az ’or die(mysql_error())’. Ez arra való, hogy hibaüzenetet küldjön, ha nem sikerül a kapcsolódás a webszerverhez.

Most nézzük meg a titkosítását a jelszónak. A jelszavak titkosítására az MD5 kivonatoló algoritmust választom. Az MD5 ugyan feltörhető, de ez jelentős időbe telik, viszont gyors, könnyen megvalósítható, és a céljaimnak megfelelően biztonságos.[8] A regisztrációnál megadunk egy string-et, amit titkosítok, és azt tárolom le az adatbázisban.

registration.php (részlet):

```
<?php
$_POST['pass'] = md5($_POST['pass']);
```

```
if (!get_magic_quotes_gpc()) {
    $_POST['pass'] = addslashes($_POST['pass']);
    $_POST['username'] = addslashes($_POST['username']);
}
$insert = "INSERT INTO ceg (username, password)
VALUES ('".$_POST['username']."', '".$_POST['pass']."')";
    $add_member = mysql_query($insert);
 ?>
```

Feltűnhetett még egy ismeretlen függvényhívás a fentebbi kódban. A `get_magic_quotes_gpc()`. Ha a magic_quotes beállítás értéke „On”, minden külső forrásból származó adat – többek között az űrlapok és a sütik – különleges bánásmódban részesül. Az idézőjel karakterek – „ és ’ – helyettesítése a fordított törtjel (\) alkalmazásával történik. Ezt eredetileg az SQL-befecskendezéses támadások megakadályozása valósították meg.[9] A kódomban ellenőrzöm, hogy be van-e kapcsolva ez az érték. Ha nincs, akkor levédem a mezőket a fordított törtjel beillesztésével. Ezután eltárolom egy változóba azt az SQL utasítást, ami beszúrja a ceg nevű táblába a felhasználónevet és jelszót. Majd erre a változóra alkalmazom a mysql_query() függvényt, ami végrehajtja a változóban letárolt SQL utasítást.

4.2.1 Sütik

A sütik (cookie) a http-fejléc részeként továbbítódnak és gyakorlatilag egy-egy név-érték párból állnak. A legfőbb hátrányuk, hogy a webböngészőben le lehet tiltani a sütiket. A sütik bizonyos korlátokkal rendelkeznek:

- A sütik tartományokhoz kötöttek, általában a sütit küldő tartományhoz.
- A sütiket a webkiszolgálón található elérési utakhoz lehet kapcsolódni.
- A sütik kizárólag szöveges információt tartalmaznak, ami legfeljebb 4096 bájt lehet.
- A böngészők tartományonként legfeljebb 20 sütit fogadhatnak, összesen pedig 300-at (néhány böngésző azonban többet).

A sütik bejegyzésekor létrejön a Set-Cookie bejegyzés a HTTP-fejlécben. Ez után a süti neve és értéke következik, valamint további, szabadon választott adatok, például a süti lejáratának ideje, a tartománya és az elérési útja.[10]

Az oldalam hitelesítéséhez sütiket használok. Bejelentkezéskor sütiben tárolom a felhasználó gépén a jelszavát és felhasználó nevét. A honlap bármely részére látogat a felhasználó rendelkeznie kell ezzel a sütivel, különben a hitelesítés nem teljesül és a bejelentkezési képernyőre dobja a user-t.

login.php (részlet):

```
<?php
$hour = time() + 3600;
setcookie(ID_username, $_POST['username'], $hour);
setcookie(Key_pass, $_POST['pass'], $hour);
echo "<meta http-equiv=\"refresh\" content=\"0;
      URL=adatlap.php\">";
?>
```

A kódban találgató time() függvény a rendszeridőt adja meg nekünk, amihez láthatóan, hozzáadunk 3600-at, ami a másodpercek számát jelenti, és ezt tároljuk a $hour változóban. A setcookie függvényben állítjuk be a 2 darab süti nevét, értékét és lejárati idejét. Ugyanis a $hour váltózó arra szolgál, hogy 1 órás munkamenetet biztosítson a dolgozók számára, aztán újra be kell majd lépni.

Mint írtam, minden egyes php állományban ellenőrizzük a sütik meglétét, így növelve a biztonságot az alkalmazásban. Részlet az ellenőrző kódból:

```
<?php
if(isset($_COOKIE['ID_username'])) {
    $username = $_COOKIE['ID_username'];
    $pass = $_COOKIE['Key_pass'];
    $check = mysql_query("SELECT * FROM ceg WHERE username =
'$username'")or die(mysql_error());
    while($info = mysql_fetch_array( $check )) {
```

```
        if ($pass != $info['password']) {
               header("Location: login.php");
        } else {
            . . .
        }
    }
 } else {
     header("Location: login.php");
}
?>
```

A fentebbi kódban először megnézzük, hogy létezik-e süti ’ID_username’ névvel. Ha létezik, akkor feltételezem, hogy létezik a ’Key_pass’ nevű süti is és ezeknek az értékeit tárolom egy-egy változóban. Egy SQL Select utasítással lekérem a username-hez tartozó minden ceg táblabeli adatot, majd megnézem, hogy a jelszavak egyeznek-e (a táblában levő és a sütiben levő). Ha nem egyeznek, akkor bejelentkezési lapra irányítom a látogatót, de ugyanez lesz a helyzet akkor is, ha nincs süti a böngészőben. A kipontozott részre mehet a php állomány többi motorikus része.

4.2.2 Felület, design

A felületet igyekeztem homogén szemkímélő színekből összeállítani. Az oldalsáv egy homogén szürke csíkos háttér, közepén világos mezővel. Ezt egy css file-ból hívom meg.

st.css (részlet):

```
    body {
     background-image: url(images/bg2.png);
     background-position: center center;
     height: 100%;
     margin: 0px;
     padding: 0px;
    }
```

A következő lépésem egy logó tervezése volt, ami igen amatőrre sikerült, de úgy gondolom, hogy barátságosabb egy honlap kinézetű alkalmazás, mert a használóját arról győzheti meg, hogy internetezés közben dolgozik. Ez a felhasználó-barátság egyik lépcsőfoka. A logót és egyéb újrafelhasználandó HTML elemeket egy html állományban gyűjtöttem össze. Ezt a html állományt is include-olom a php file-okban.

alap.html tartalma:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> Webacar </TITLE>
<META NAME="Author" CONTENT="Székely Szilárd">
<META NAME="Keywords" CONTENT="szakdolgozat, webacar,
betegszállítás">
<META NAME="Description" CONTENT="Betegszállítói rendszer">
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=UTF-8">
<link href="st.css" rel="stylesheet" type="text/css">
<script type="text/javascript" src="images/jquery-
1.4.2.min.js"></script>
<script type="text/javascript"
src="images/jquery.color.js"></script>
<script type="text/javascript">
$(document).ready(function() {
 $('#menu-jquery li a').hover(
  function() {
    $(this).css('padding', '5px 15px')
     .animate({'paddingLeft'  : '25px', 'paddingRight' :'25px',
          'backgroundColor':'rgba(0,0,0,0.5)'}, 'fast');},
  function() {
    $(this).css('padding', '5px 25px')
```

```
     .animate({'paddingLeft'  : '15px', 'paddingRight' :’15px',
          'backgroundColor' :'rgba(0,0,0,0.2)'}, 'fast');})
     .mousedown(function() {
          $(this).animate({'backgroundColor':
     'rgba(0,0,0,0.1)'}, 'fast');}).mouseup(function() {
          $(this).animate({'backgroundColor':
     'rgba(0,0,0,0.5)'}, 'fast');});
});
</script>
</HEAD>
<body>
<p align="center"><img src="images/logo.png"/></p>
<p align="center">
<ul id="menu-jquery">
     <li><a href="adatlap.php">Cég információk</a></li>
     <li><a href="auto_lista.php">Autók</a></li>
     <li><a href="korhaz_lista.php">Kórházak</a></li>
     <li><a href="orvos_lista.php">Orvosok</a></li>
     <li><a href="beteg_lista.php">Betegek</a></li>
     <li><a href="betegszall.php">Adatlapok</a></li>
     <li><a href="dolgozo_lista.php">Dolgozók</a></li>
     <li><a href="kimutatas.php">Kimutatás</a></li>
     <li><a href="logout.php">Kilépés</a></li>
</ul>
</p>
```

st.css (részlet):

```
ul {
     text-align: center;
}

#menu-jquery li {
```

```
     display: inline;
     list-style: none;
}

#menu-jquery li a {

/* Határoló sugár */
-webkit-border-radius: 15px;
-moz-border-radius: 15px;
border-radius: 15px;

/* Határoló árnyék */
-webkit-box-shadow: 1px 2px 2px rgba(0,0,0,0.6);
-moz-box-shadow: 1px 2px 2px rgba(0,0,0,0.6);
box-shadow: 1px 2px 2px rgba(0,0,0,0.6);

color: #ffffff;
background: rgba(0,0,0,0.2);
display: inline-block;
padding: 5px 15px;
outline: none;
text-decoration: none;
}

#menu-jquery li a:active {
     -webkit-box-shadow: 1px 1px 1px rgba(0,0,0,0.4);
     -moz-box-shadow: 1px 1px 1px rgba(0,0,0,0.4);
     box-shadow: 1px 1px 1px rgba(0,0,0,0.4);
}

a, a:visited {
     color: #161616;
     text-decoration: none;
```

```
}
```

A HTML kódot a szokásos sorral kezdem, amit az EditPlus3 automatikusan elkészít. Ezután a fejrész következik. Itt adom meg a honlap címét, ami a böngészők címsora lesz, majd meta-adatokkal írom le a szerzőt és egyéb kulcsszavakat. A CSS file-t és az előre megírt JavaScript-et is itt importálom. A JavaScript-et, itt a jQuery-hez szükséges állomány tartalmazza. Nem beimportált JavaScript is van a kódban, mely a menühöz szükséges. Úgy gondoltam, ahhoz, hogy még jobban megnyerjem a felhasználót magamnak egy animált menüt szeretnék készíteni. De a flash-t szerettem volna hanyagolni, helyette egy ma igazán népszerű nyelv (jQuery) segítségével készítem el a menüt. Mint, ahogy a jQuery bemutatásánál írtam, ez a nyelv azonosítókon keresztül éri el HTML elemeket. Jelen esetben a css-t és a body-ban található lista elemet, mely el van látva egy „menu-jquery” azonosítóval. A JavaScript-ben hivatkozok a css-re, melyben a megjelenítési információkat tárolom az azonosítóval ellátott listával kapcsolatban. Itt adom meg a színeket, árnyékokat, a gombok görbületét.

Az rgba függvényhívás lehetőséget ad arra, hogy alfa átlátszóságot használjak a rendszerben. Paraméterei rendre: piros, zöld, kék valamint az alfa átlátszóság értékei. Bizonyos böngészők ezt még nem támogatják [11]. A továbbiakban pedig az egér mozgására figyel a script és ennek megfelelően változtatja, animálja a menülista elemeit.

4.3 Biztonság

Minden egyes php állományban, úgy alakítom ki a szerkezetet, hogy az POST-olt adatokat ne másik php állomány kezelje, hanem saját maga. Ezt azért így oldom meg, mert így a böngésző url mezőjében nem lehet a paramétereket átállítani, azaz nem férnek hozzá illetéktelen felhasználók.

Példa az űrlapokra a programban, mely önmagának küldi el az adatokat:

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
method="post"> . . .
</form>
```

További biztonsági intézkedéseim a sütiknél leírásra került, hogy hogyan hitelesítek egy felhasználót. A bejelentkezésnél pedig az SQL befecskendezéses támadást próbálom kiszűrni, de már erről is beszámoltam korábban. Tovább lehet erősíteni az alkalmazást, ha a szerver oldalról is megteszünk minden óvintézkedést. Ez a MySQL, Apache és PHP beállításainak a finomhangolását jelenti. Erre nincs túl sok lehetőségem egy ingyenes kiszolgálónál, meg kell, hogy elégedjek jelenleg azzal, hogy akik ezt a szolgáltatást kínálják, bizonyára figyeltek ilyesmire.

5. Üzleti logika megvalósítása, avagy áttekintés

A legfontosabb, hogy alkalmazásom megfelelően működjön az elvárásoknak. Az elvárásokat a betegszállítói cég intézte felém. A dolgozatom ezen részében szeretném összefoglalni, hogy mik voltak a kérések.

Az interjún megtekinthettem a jelenlegi alkalmazást, mely egy DOS-os felületű program volt rengeteg funkcióval. Első kérdésem arra irányult, hogy milyen funkciókat használnak abban a szoftverben. Az elsődleges az, hogy a betegszállítási adatlapról rögzítsék a szállított betegeket, a szállító autó kilométerállását, a megrendelő intézményt, a megrendelő orvost, és az ezekhez tartozó adatokat. Ez alatt a kórházak nevét, ÁNTSZ kódját, az orvosok pecsétszámát jelentik. De csak azokat kellett rögzíteniük, amik nem voltak benne az alkalmazásban.

Mialatt figyeltem a régi program működését használat közben, azt találtam, hogy rendkívül gyors a Windows XP környezetben, lévén, hogy egy DOS-os alkalmazásnak nem sok erőforrásigénye van. Itt már megfordult a fejemben egy gondolat, amit később elhatározás követett: Nem alkalmazhatok olyan új technológiákat, aminek nagyobb erőforrásigénye van, és rengeteg file-ból dolgozik. Ugyanis az első gondolatom az volt, hogy a GWT (Google Web Toolkit)-t elsajátítva valósítom meg webalkalmazásom. Erről azonban le kellett tennem, mert többszöri próbálkozás után, nem találtam elég gyorsnak ezt a fejlesztői eszköztárat.

Láttam azt is, hogy az adatrögzítés egy laptopon történik. Kérdeztem, hogy mi van akkor, ha valami okból kifolyólag ez a gép nem indul be többet. A válasz az volt, hogy akkor gubanc van. Itt egy kicsit megkönnyebbültem, hogy máris van a webalkalmazásomnak több előnye is, mégpedig, hogy az adatrögzítés bármilyen gépről elvégezhető, ahol működő internetkapcsolat van és nem egy géphez vagyunk kötve. A másik előnye az új rendszernek, hogy tanulóképes, mint az elődje, csak ebben nem csak a saját adatbázissal lesz okosabb a rendszer, hanem mindem olyan betegszállító cég adataival, aki használja majd a webes programot. Itt most a kórházak és az orvosok adatbázisára gondolok.

Következő kérdésem az irányba szólt, hogy mit kell még tudnia a rögzítésen túl ennek a DOS-os kis programnak. Nyomtatni. Csakhogy valamilyen áthidaló megoldással lehetett

kinyomtatni olyan információkat, amiket végül postázni kellett valamilyen központi szervezetnek. Az áthidaló megoldás eléggé bonyolultnak tűnt számomra, és ráadásul floppy-t is kellett használni. Milyen információkat kell kinyomtatni? Az adatbázis szerkezetnél tárgyaltam ezt a részt, miszerint egyedi és kapcsolt szállításokat megkülönböztetve kell a betegszállítói autó megtett kilométerét egy adott időintervallumon megadni. Gondoltam ezt a funkciót a legegyszerűbben oldom meg. Ha webes alkalmazásról beszélünk, használhatjuk a böngészőnk beépített tulajdonságait is. Tehát arra gondoltam, hogyha meglesznek az adatok, akkor egy design nélküli lapra az adatokat kiíratva, majd a böngésző nyomtatás menüpontját használva lényegesen egyszerűbb a nyomtatás.

5.1 Rendszerkövetelmények:

- Internet hozzáférés, böngésző
- Windows, GNU/Linux, MacOS
- a várható felhasználó cégek száma kb. 40-50 országosan

5.2 Fogalomszótár:

- **Admin** – Az a személy, aki regisztrálhat a rendszerbe új felhasználót, valamint törölhet orvost és kórházat az egész rendszerből.
- **Regisztrált felhasználó –** adatokat rögzíthet a rendszerben, valamint néhol törölhet és módosíthat adatain.
- **Beteg** – Az a személy, aki igénybe veszi a betegszállítói cég szolgáltatását, így egy kiindulási pontból egy végpontba szállítandó. Adatai rögzítésre kerülnek.
- **Orvos** – Az a személy, aki berendeli vagy átveszi a beteget a kórházban. Adatai rögzítésre kerülnek.
- **Kórház** – Az az intézmény ahol a betegeket kezelik, ápolják. A betegszállító program szempontjából lehet kiindulási pont vagy végpont. Adatai rögzítésre kerülnek.
- **Autó** – A betegszállítói autót kell ezalatt érteni, amivel a szállítás történik. Adatai rögzítésre kerülnek, valamint frissülnek.

- **Dolgozó** – Egy betegszállítói cégnél dolgozó ember. Lehet akármilyen beosztásban: ápoló, sofőr, titkár, stb.
- **Kimutatás** – Egy kimutatás két adott időpont között feldolgozandó adatot jelenti.
- **Cég** – Maga a betegszállítói cég.
- **Adatlap** – A betegszállítói adatlap rövidítése. Ez egy formanyomtatvány elektronikus megfelelője.

5.3 Szakterületi kapcsolatok és folyamatok:

- **Autó felvétele** – Az a folyamat, mikor a cég felveszi az adatbázisába az összes betegszállítói autót. Egy autónak van rendszáma, típusa, márkája, forgalmi érvényessége, kiadási évjárata, férőhelyszáma és kilométerállása. Céghez köthető.
- **Autó törlése** – Az a folyamat, mikor egy autót nem használ tovább a cég, és szeretné törölni. Csak logikai törlés valósul meg.
- **Kórház felvétele** – Mikor egy cég olyan egészségügyi intézményt regisztrál, ami még nincs az adatbázisban. Nem köthető céghez. Globális adat.
- **Kórház törlése** – A kórházat ilyenkor fizikailag töröljük az adatbázisból. Csak admin felhasználó törölhet.
- **Orvos felvétele** - Mikor egy cég olyan orvost regisztrál, aki még nincs az adatbázisban. Nem köthető céghez. Globális adat.
- **Orvos törlése** – Az orvost ilyenkor fizikailag töröljük az adatbázisból. Csak admin felhasználó törölhet.
- **Beteg felvétele** - Mikor egy cég olyan beteget regisztrál, aki még nincs az adatbázisban. Céghez köthető.
- **Beteg törlése** – Egy cég felhasználója logikailag törölhet egy beteget.
- **Dolgozó felvétele** - Mikor egy cég olyan dolgozót regisztrál, aki még nincs az adatbázisban. Céghez köthető.
- **Dolgozó törlése** - Egy cég felhasználója logikailag törölhet egy dolgozót.
- **Adatlap felvétele** - A nyomtatott betegszállítási adatlap elektronikus rögzítése a cég által. Céghez köthető.
- **Kimutatás készítése** – Adott időintervallumból kiszámolt kilométer megmutatása és nyomtatási lehetősége.

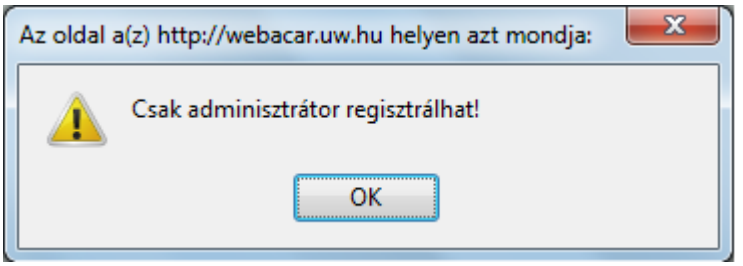
6. Program bemutatása képernyőképekkel

A bejelentkezés:

A login.php meghívása ezen oldal létrejöttét eredményezi, ha nincs olyan süti eltárolva a gépünkön, amely az oldallal kapcsolatos adatokat tartalmazhat. Ha elgépeljük a felhasználónevünkhöz tartozó jelszót, akkor a rendszer egy „Rossz felhasználónév vagy jelszó” üzenetet jelenít meg. Ha olyan felhasználóval szeretnénk belépni a rendszerbe, amely nem létezik, ekkor a program azt az üzenetet dobja, hogy „Nincs ilyen felhasználónév vagy jelszó. Kérjük, regisztráljon!”. A linkre kattintva a regisztrációs oldalon találja magát a látogató.

Regisztrációs oldal:

Ha bármilyen külsős cég próbál regisztrálni, azt a rendszer nem fogja engedni, ugyanis csak „admin” jogokkal lesz lehetséges ez a művelet.



Belépés után a következő képernyő fogad, amelyen megjelenik egy felső menüsor és a regisztrált tag cégének adatai. Ez az adatlap.php:

Erről az oldalról lehetőség nyílik a céges adatok módosítására, valamint a menünek a többi ágába való eljutáshoz.

Minden egyes olyan menüpont, melyben információt, adatot kérünk le a szerverről, egységes kinézetet kapott. Kivéve az Autók menüpont:

Ilyen a Kórházak, Orvosok, Betegek, Dolgozók menüpont. A listázás egy táblázatba került a jobb átláthatóság kedvéért.

Törlés:

Lehetősége van egy normál felhasználónak logikai törlésre, melyet a betegein, autóin és a dolgozóin végezhet el. Itt szintén kapunk egy felsorolást, és minden egyes rekordhoz tartozik egy checkbox. Akár több rekordot is tud egyszerre törölni a program használója.

Adat felvételekor az Autók, a Kórházak, az Orvosok, a Betegek és Dolgozók szintén egységes egészet alkotnak a felület tekintetében.

Minden egyes beviteli mező mellé zárójelben egy példa szemlélteti a helyes beviteli formát.

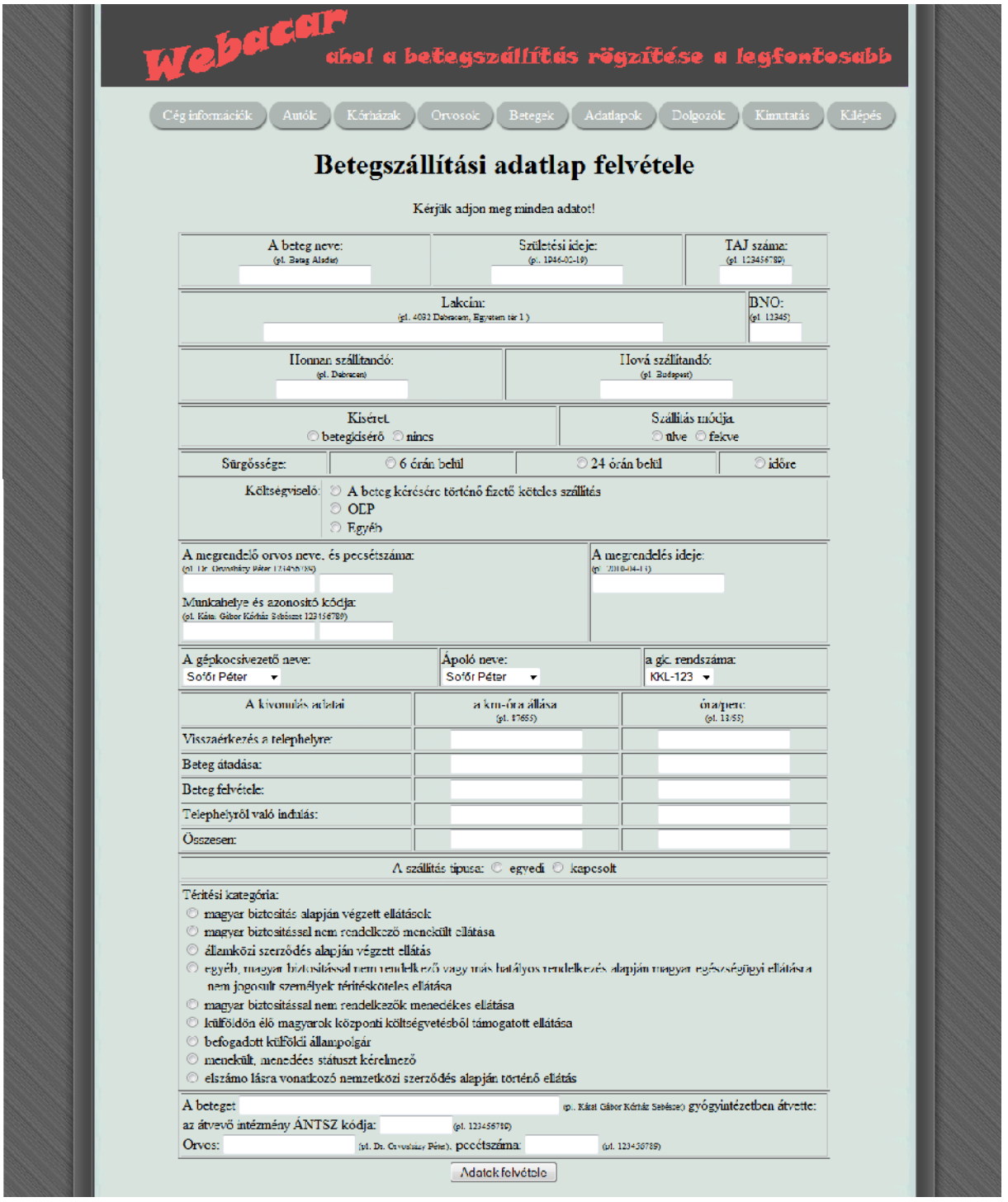
Ha kimutatást szeretnénk készíteni, akkor csak annyi a teendő, hogy egy kezdeti és egy végső időpontot határozzunk meg. Ezután a ’Mutasd’ gombra kattintva egy Tiszta oldalra leszünk irányítva:

Kimutatás

Cég	A 2010-01-01 és a 2010-12-31 közötti km eredmények:
Fiktív Cég Kft.	Egyedi km: 0 Képesolt km: 100 Összes km: 100

Ezután a böngésző nyomtatás funkciójára kattintva gond nélkül és nyomtatóbarát módon ugyanazt megvalósíthatjuk, mint eddig a régi programmal, csak sokkal könnyebben.

A legfontosabb és legösszetettebb felület a betegszall.php által előállított felület. Mint azt látni lehet, itt is a beviteli mezők felett, (alatt, mellett) példa szolgál az adatok helyes megadása érdekében. Maga a szerkezet pedig a formanyomtatványt követi, így a bevitel ideje tovább rövidül. Sok helyen, csak radio-gombokat használtam. Minden radio-gomb csoportokba van rendezve, így egy-egy csoportból csak egyet lehet megjelölni. A sofőr és az ápoló a dolgozok táblából kerül feltöltésre, így egy legördülő menüből lehet őket kiválasztani. Hasonlóképp működik a betegszállító autó rendszámának megadása is.


Webacar
ahol a betegszállítás rögzítése a legfontosabb
Cég információk
Autók
Kórházak
Orvosok
Betegek
Adatlapok
Dolgozók
Kimutatás
Kilépés
Betegszállítási adatlap felvétele
Kérjük adjon meg minden adatot!
A beteg neve:
(pl. Beteg Aladár)
Születési ideje:
(p.. 1946-02-19)
TAJ száma:
(pl. 123456789)
Lakcím:
(pl. 4032 Debrecen, Egyetem tér 1)
BNO:
(pl. 12345)
Honnan szállítandó:
(pl. Debrecen)
Hová szállítandó:
(pl. Budapest)
Kíséret
betegkísérő
nincs
Szállítás módja
ülve
fekve
Sürgőssége:
6 órán belül
24 órán belül
időre
Költségviselő:
A beteg kérésére történő fizető köteles szállítás
OEP
Egyéb
A megrendelő orvos neve, és pecsétszáma:
(pl. Dr. Orvosházy Péter 123456789)
A megrendelés ideje:
(p. 2010-04-14)
Munkahelye és azonosító kódja:
(pl. Kása. Gábor Kórház Sebészet 123456789)
A gépkocsivezető neve:
Sofőr Péter
Ápoló neve:
Sofőr Péter
a gk. rendszáma:
KKL-123
A kivonulás adatai
a km-óra állása
(pl. 17655)
óra/perc
(pl. 13:55)
Visszaérkezés a telephelyre:
Beteg átadása:
Beteg felvétele:
Telephelyről való indulás:
Összesen:
A szállítás típusa: egyedi kapcsolt
Térítési kategória:
magyar biztosítás alapján végzett ellátások
magyar biztosítással nem rendelkező menekült ellátása
államközi szerződés alapján végzett ellátás
egyéb, magyar biztosítással nem rendelkező vagy más hatályos rendelkezés alapján magyar egészségügyi ellátásra nem jogosult személyek térítésköteles ellátása
magyar biztosítással nem rendelkezők menedékes ellátása
külföldön élő magyarok központi költségvetésből támogatott ellátása
befogadott külföldi állampolgár
menekült, menedékes státuszt kérelmező
elszámo lásra vonatkozó nemzetközi szerződés alapján történő ellátás
A beteget
(p.. Kásá Gábor Kórház Sebésze)
gyógyintézetben átvette:
az átvevő intézmény ÁNTSZ kódja:
(pl. 123456789)
Orvos:
(pl. Dr. Orvosházy Péter),
pecétszáma:
(pl. 123456789)
Adatok felvétele

7. Összefoglalás:

Úgy gondolom a dolgozatomban sikerült elérni azt a célt, hogy adatrögzítésre és kimutatás készítésére alkalmas a programom, melyet egyszerre több felhasználó használhat. Igyekeztem arra törekedni, hogy ha újabb igények merülnek fel egy betegszállítói cég felől, akkor könnyen továbbfejleszthető legyen az alkalmazás. Úgy érzem, hogy nagyon alap funkciókkal rendelkezik a Webacar, így éles bevetés előtt mindenképp javítani kell rajta és több funkcióval kell ellátni. Azt a célt viszont nem sikerült elérnem, hogy a Doctrine keretrendszert is használjam az alkalmazásban. Ha továbbfejlesztésre kerül a sor, ez mindképp belekerül.

Irodalomjegyzék:

- [1] Rendszerfejlesztés technológiája jegyzet
- [2] http://hu.wikipedia.org/wiki/MySQL
- [3] http://hu.wikipedia.org/wiki/PHP
- [4] http://hu.wikipedia.org/wiki/Apache_HTTP_Server
- [5] http://hu.wikipedia.org/wiki/JavaScript
- [6] http://hu.wikipedia.org/wiki/JQuery
- [7] http://hu.wikipedia.org/wiki/Html
- [8] TRICIA BALLAD & WILLIAM BALLAD – Biztonságos Webalkalmazások PHP nyelven 128.old (Kiskapu 2010)
- [9] CHRISTIAN WENZ – PHP zsebkönyv 100.old (Kiskapu 2007)
- [10] CHRISTIAN WENZ – PHP zsebkönyv 138-139.old (Kiskapu 2007)
- [11] http://css-tricks.com/rgba-browser-support/
- http://php.about.com/od/finishedphp1/ss/php_login_code.htm
- „how to” kezdetű keresések a google-ban. pl.: how to create animated menu with jquery and css

Köszönetnyilvánítás:

- Dr. Kuki Attilának, hogy elvállata a témavezetőm szerepét és segítségéért, hogy javaslataival emelte a dolgozatom színvonalát.
- Unokatestvéremnek, hogy beavatott a betegszállításba.
- Menyasszonyomnak, hogy tartotta bennem a lelket és ösztönzően hatott rám.
- Szüleimnek és rokonaimnak a támogatásukért.