DIPLOMAMUNKA

Oláh Zsolt

Debrecen
2010

Debreceni Egyetem

Informatika Kar

Webes adminisztrációs felület laborkörnyezet távoli hozzáférésének időzítéséhez

Témavezető:

Dr. Almási Béla
egyetemi docens

Készítette:

Oláh Zsolt
programtervező matematikus
szakos hallgató

Debrecen
2010

Tartalomjegyzék

Bevezetés

Diplomamunkám során gyakorlati igényt kívánok kielégíteni. A Debreceni Egyetemen meghirdetett Cisco hálózati alapok 1 és Cisco hálózati alapok 2 című tárgyak sikeres elvégzését támogató, a Cisco Akadémiai környezetek laboreszközökhöz távolról való hozzáférést biztosító HLAB rendszert egészítem ki további funkcionalitással, nevezetesen webes adminisztrációs felületet készítek a laboreszközök távoli elérésének időzítéséhez. Az időzítés maga már eddig is megoldott probléma konzolos felületen, de jelenleg ehhez egy adminisztrátor munkája szükséges, mert az oktatók nem rendelkeznek információval arról, hogy mely időszakokat foglalták le már más oktatók. Kitűzött célom az, hogy kényelmes webes felületen keresztül áttekintést adjak a foglalásokról, így tehermentesítve a HLAB rendszeradminisztrátort, és az oktatók önállóan végezhessék el a laboreszköz-foglalásokat.

Az elkészítendő felület tehát egy már meglévő működő rendszer kiegészítése, annak funkcionalitását és kényelmességét növelő eszköz lesz. Ez természetesen azt jelenti, hogy a meglévő rendszer minden korábbi funkciója továbbra is elérhető marad a megszokott módon.

A feladat megoldásához úgynevezett keretrendszert veszek igénybe, amely sok automatizálható rutinfeladat alól mentesít, így valóban az elvégzendő feladatra koncentrálhatok Választásom olyan keretrendszerre esett, amely az **MVC** (Model-View-Controller) tervezési mintán alapul. Az MVC felépítés a program adatainak, megjelenésének és vezérlésének szétválasztását eredményezi. Ez mind a fejlesztést mind az esetleges módosítást, továbbfejlesztést megkönnyíti. Mivel a már meglévő alaprendszer Linux alapon fut, a választott keretrendszernek szintén működnie kell ezen a platformon. Választásom a szabadon letölthető **Catalyst** keretrendszerre esett, amelynek programozási nyelve a Perl szkriptnyelv.

A fejlesztést természetesen nem az éles HLAB rendszeren végzem, hanem az éles rendszerhez hasonló fejlesztési környezetben. A fejlesztési környezet HLAB számára előkonfigurált Linux alaprendszerből, további telepített alkalmazásokból (böngésző, integrált fejlesztői környezet), és a Catalyst keretrendszerből áll. E környezet nem külön fizikai számítógépen fut, hanem virtuális gépen, amely biztonságos és kis költségű fejlesztést tesz lehetővé. Biztonságos, mert a virtuális gép megfelelően elkülönül attól a fizikai számítógéptől, amelyen fut; valamint kis

költségű, mivel a választott virtualizációs szoftver (a virtuális gép) ingyenesen használható, illetve nincs szükség másik, dedikált számítógép beszerzésére.

Különösen törekszem arra, hogy az elkészült rendszer futtatása ne okozzon jelentős többletterheléssel a HLAB rendszernek otthont adó hardver- és szoftverkörnyezet számára, hogy csak a feltétlenül szükséges erőforrásokat használja.

Elmondható, hogy korszerű technológiákat alkalmazok a kitűzött feladat megoldásához, hiszen a komolyabb fejlesztéseket ma már keretrendszerekben végzik; illetve a virtualizáció is igen népszerű manapság.

A szoftverfejlesztést szintén megkönnyíti az úgynevezett verziókezelő rendszerek használata, amelyek segítségével a csapatmunka is egyszerűbbé válik, megkönnyítik a különböző fejlesztési változatok (verziók) közötti eligazodást, lehetővé teszik a korábbi változatokhoz való visszatérést, a fejlesztési elágazások használatát. Ilyen verziókezelő rendszer például a régebbi **CVS** (Concurrent Versions System), vagy az újabb **SVN** (Subversion). Az SVN kiegészítve a **trac** projektmenedzsment rendszerrel pedig igen jól kezelhetővé teszi a projektmunkát. Ez webes felületet ad az SVN-hez, tartalmaz beépített Wiki rendszert, és sok más hasznos eszközt. Ebben a projektben lehetett volna ugyan verziókövető rendszer nélkül is dolgozni, mert a projekt kisméretű, és nem csapatmunkában készül. Mégis úgy gondoltam, hogy hasznos lesz kicsit megismerkedni az SVN-nel a gyakorlatban.

Most pedig arról essék szó, milyen területeket is fogok diplomamunkám során tárgyalni. Először bemutatom a fejlesztési környezet kialakításának lépéseit, majd tervezési megfontolások következnek. A tervezés után pedig a tényleges programfejlesztésre kerül sor. Telepítési csomagot is készítek az elkészült rendszer minél kényelmesebb használatba vétele érdekében, valamint felhasználói dokumentáció is készül a fejlesztett felülethez. A függelékben közlöm a legfontosabb rövidítéseket, és azok feloldásait, magyarázattal. Az összefoglalásban pedig megnézem, sikerült-e elérni a kitűzött célokat mindegyikét. Ha esetleg mégsem, akkor pedig megkísérelem feltárni az okokat.

1. A fejlesztési környezet kialakítása

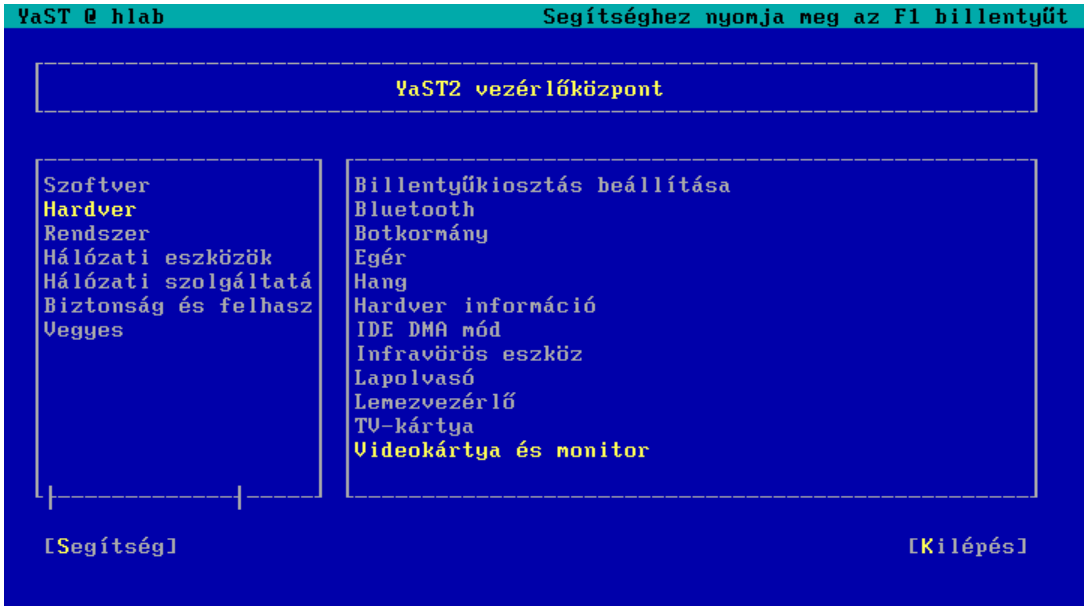
A fejlesztési környezet alapja az openSUSE 10.1 Linux operációs rendszer. Ehhez jön a HLAB alapkonfiguráció. Ezeket két ISO CD-lemezkép segítségével telepítettem, amelyeket témavezetőm bocsátott rendelkezésemre, a telepítési útmutatóval egyetemben. Az általam választott virtuális gép képes CD-lemezként használni az ISO CD-lemezképeket, így nem volt szükséges lemezre írnom őket a telepítéshez. A telepítés ezen szakaszát nem részletezem, csupán annyit említek meg, hogy a minimálisan ajánlott 256 MB memória helyett 512 MB-ot adtam a virtuális gépnek, illetve 3 GB lemezméret helyett 20 GB-ot, hogy a fejlesztést ne korlátozzák ezek a paraméterek. Természetesen a rendszer működéséhez már nem szükséges ekkora erőforrás. A továbbiakban arról lesz szó, hogyan is állt elő a fejlesztési környezet.

1.1. Grafikus felület beállítása

A fejlesztéshez nem feltétlenül szükséges grafikus felhasználói felület, hiszen elegendő lenne a konzol, illetve egy szövegszerkesztő program (például a **vi**, amely lényegében minden Linux terjesztésben megtalálható, vagy legalábbis annak valamely továbbfejlesztett változata mindenképpen). Mégis, a célszerűség és a kényelem miatt integrált fejlesztői környezetet (Integrated Development Environment, **IDE**) alkalmaztam munkám során, amelynek szüksége van működő grafikus környezetre.

A grafikus felület beállításához az openSUSE menüvezérelt beállító-, konfiguráló-, telepítő- és rendszeradminisztrációs programját, a **YaST** (Yet another Setup Tool) nevű eszközt használtam amely a `yast` parancs kiadásával futtatható, *root* felhasználóként (Pontosabban, a `/sbin/yast` parancs kiadásával korlátozott felhasználóként is elindíthatjuk a YaST-ot, de nem lesz jogosultságunk a grafikus beállítások elvégzéséhez). A legfontosabb irányító billentyűk a YaST-ban a következők: Nyilak (mozgás a menük között illetve a menükben), Tabulátor (mozgás az egyes képernyőelemek között, például a főmenü és az almenük között), Enter (kijelölt elembe, pl. almenübe lépés).

A YaST az operációs rendszer telepítésekor megadott nyelven jelenik meg, ez most a magyar. A következő képernyőképen (1. ábra) látható, hogy a „Hardver” menüpont „Videokártya és monitor” almenüjét választottam ki, ezzel végezhető el a grafikus felület beállítása. Enter ütöttem a kijelölt almenün, és kicsit várnom kellett a grafikus beállítóprogramra.

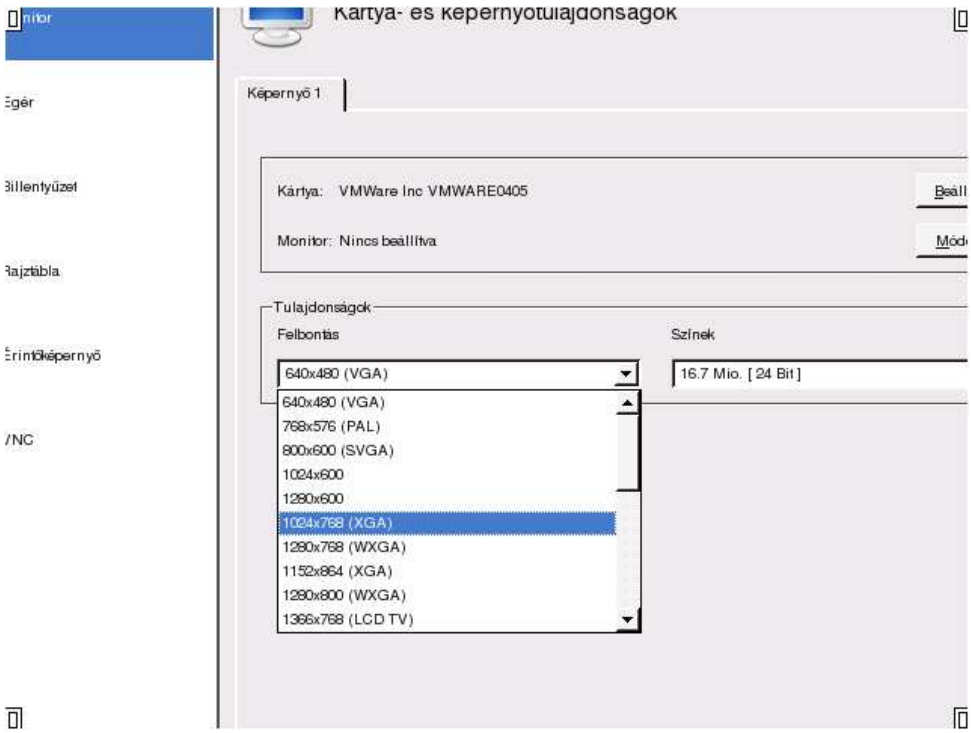


1. ábra

Amint a 2. ábrán láthatjuk, a program felajánl egy automatikus beállítást. Ezt elfogadhatjuk, vagy pedig a „Beállítások módosítása” nyomógomb segítségével egyéni beállítást alkalmazhatunk. Mivel a felajánlott felbontást kicsinek találtam a kényelmes fejlesztési munkához, a „Beállítások módosítása” gombra kattintván kézzel állítottam be a grafikus felületet.

2. ábra

Nos, valójában még a beállítóprogram számára is kicsinek bizonyult a felajánlott felbontás, az ablak egy része kilóg a képernyőből. A „Tulajdonságok” alatt a felbontást 1024×768 képpontra változtattam (megítélésem szerint ekkora felbontás már elegendő lesz a hatékony munkához).

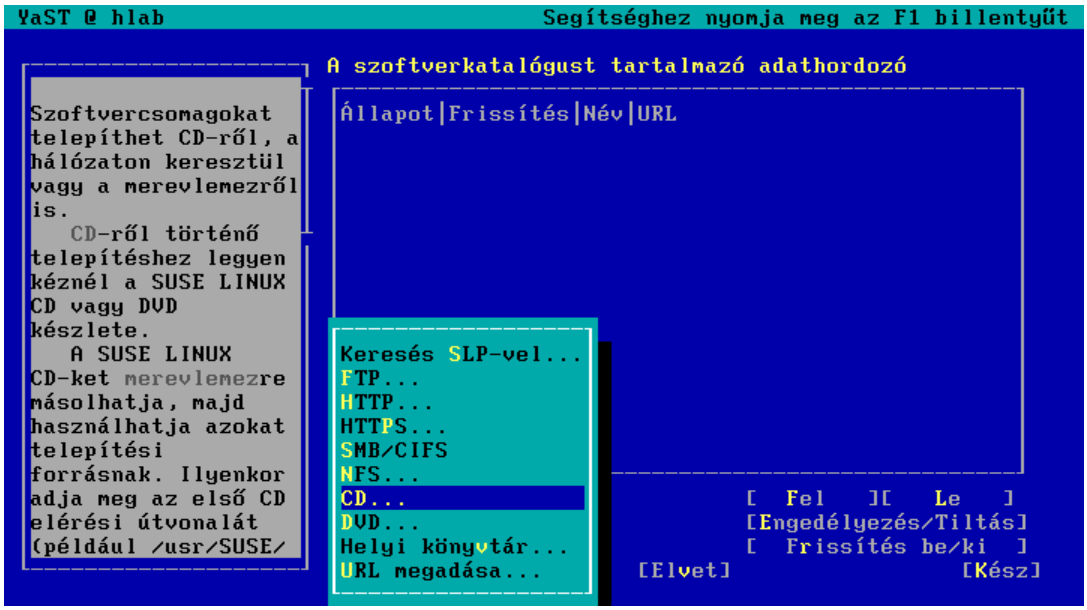


3. ábra

Mivel nem látszott az „OK” gomb, kicsit trükköznöm kellett a Tab billentyűvel. Három Tab után ütöttem egy Entert, amire előjött egy figyelmeztető ablak, hogy mentés előtt érdemes tesztelni az új grafikus beállításokat. Ezt meg is tettem, hogy elkerüljem a későbbi kellemetlenségeket. Ilyen lehet például az operációs rendszer induláskor lefagyó grafikus bejelentkező rendszer okozta patthelyzet. Mivel a beállítások megfelelőnek bizonyultak, elmentettem őket és kiléptem a grafikus beállítóprogramból (itt még kaptam egy figyelmeztetést, hogy az új beállítások csak a grafikus rendszer újraindítását követően lépnek életbe). Ezzel a grafikus beállításokat sikeresen el is végeztem.

1.2. További telepítések

A fejlesztéshez szükséges még további szoftverek (asztali környezet, böngésző, a fejlesztői környezet, a verziókövető rendszer, illetve a keretrendszer) telepítése. A telepítésekhez ismét elővettem a YaST-ot. A „Szoftver” főmenü „Telepítési forrás meghatározása” almenüjével kezdtem. Itt határozhatjuk meg, hogy milyen forrás(ok)ról kívánunk telepíteni a YaST segítségével. Egyetlen forrásként az openSUSE 10.1 telepítő DVD-jét állítottam be. Furcsa módon a hozzáadásnál CD-t kellett kiválasztanom, ha DVD-t adtam meg, akkor nem működött dolog.



4. ábra

Ezek után már telepíthetünk is a „Szoftver telepítése, eltávolítása” almenü segítségével.

Először a **GNOME** asztali környezetet jelöltem telepítésre, mert ezt sokkal kényelmesebbnek találtam a már telepített **fvwm**-nél. A „Szűrő” legördülő listából kiválasztottam az „Összeállítások” pontot. A csomagkategóriák között megkerestem a „GNOME rendszer”-t és a Szóköz billentyűvel kiválasztottam telepítésre.

Ezután a **Mozilla Firefox** böngészőt kerestem meg a „Szűrő” legördülő lista „Keresés” pontja segítségével. Keresőfeltételként a „firefox” karaktersorozatot adtam meg (természetesen az idézőjelek nélkül). Találtam is három csomagot, amelyek mindegyike ki volt már jelölve telepítésre (vélhetően a GNOME asztali környezet telepítéséhez szükséges e három csomag,

ezért voltak telepítésre jelölve). A Firefox mellett telepítésre jelöltem a **lynx** nevű karakteres böngészőt is, amellyel a konzolból is böngészhetünk (A lynx meglétét a Catalyst keretrendszer is feltételezi). A böngészők egyrészt a további lényeges elemek kényelmes letöltéséhez, másrészt a fejlesztett webes felület teszteléséhez szükségesek.

Az SVN verziókövető rendszerhez majdnem minden csomag telepítve volt már, kivéve a **subversion-server** csomagot, amelyet így szintén telepítésre jelöltem.

A tényleges telepítést az „Elfogadás” feliratú gombbal indítottam el. Elfogadtam a végfelhasználói licensz-megállapodásokat (több csomaghoz is volt). A YaST értesített azokról a változásokról, amelyek a kézzel kiválasztott csomagon felül további csomagokat érintett, a függőségek feloldása miatt. Ezt tudomásul vettem az „OK” gombbal. Ezután pedig várakoztam a telepítés befejeződéséig. A telepítési folyamat aktuális állapota szépen követhető volt: Mennyi csomag telepítése van még hátra, és ez mennyi időt vesz igénybe.

Integrált fejlesztői környezetnek az **EasyEclipse for LAMP** szoftvercsomagot választottam, amely az **Open Software License 2.1** licensz keretében szabadon használható. E csomag minden olyan eszközt tartalmaz, amelyre szükségem lehet a fejlesztés során. A következőkben az EasyEclipse telepítésének menetéről lesz szó.

A feltelepített GNOME asztali környezetet a rendszergazdaként kiadott `gdm` (GNOME Display Manager) paranccsal indítottam el, majd bejelentkeztem a *hlab* nevű korlátozott felhasználóval, mert nem tanácsos rendszergazdaként belépni grafikus felületre, hiszen akkor minden indított program rendszergazdai jogosultságokkal fog futni, amely veszélyt jelenthet a rendszer számára. Megkerestem az „Applications” menüben az „Internet/Web Browser” almenüben a Firefox böngészőt, készítettem hozzá parancsikont az asztalra, hogy mindig kéznél legyen majd. Ezután elindítottam a böngészőt és letöltöttem a http://www.easyeclipse.org/site/distributions/index.html oldalon található, a *Distributions for web and dynamic languages* szakaszban szereplő *EasyEclipse for LAMP* Linux rendszerekre írt változatát. Nyitottam egy konzolt az „Applications” menü „System/Terminal” almenüjéből. A `su` paranccsal rendszergazdává változtam a rendszergazdai jelszó megadását követően, majd az `mc` paranccsal indítottam egy **Midnight Commander-**t. Megkerestem az előbb letöltött `easyeclipse-lamp-1.2.2.2.tar.gz` tömörített állományt a `hlab/Desktop` könyvtárban, és Enterrel beleléptem. A másik panelen

beléptem a `/opt` könyvtárba (ide szokás telepíteni a további alkalmazásokat), majd visszalépve a bal oldali panelbe, az egész könyvtárat átmásoltam a jobb panelen lévő célkönyvtárba. Ez lényegében a tömörített állomány kicsomagolását jelentette (Persze, ezt megtehettem volna a `tar xzf easyeclipse-lamp-1.2.2.2.tar.gz` parancs kiadásával is, miután a `/opt` könyvtárba másoltam a tömörített állományt). Ezután az F10 billentyűvel kiléptem a Midnight Commander-ből, valamint az `exit` parancs kiadásával visszaváltoztam a *hlab* felhasználóvá, majd pedig elindítottam a fejlesztőkörnyezetet a `/opt/easyeclipse-lamp-1.2.2.2/eclipse` parancs segítségével. Mindenféle hibajelzés nélkül elindult. Elfogadtam a felajánlott alapértelmezett *workspace*-t, azaz azt a könyvtárat, amelyben az EasyEclipse tárolja az egyes projektekhez tartozó állományokat. A „Window" menü „Open Perspective/Other" pontjával kiválasztottam a Perl nézetet, hiszen a fejlesztés Perl nyelven folyik majd. Végezetül készítettem parancsikont az asztalra az EasyEclipse-hez, a gyorsabb indíthatóság érdekében.

Ezek után következzék a Catalyst keretrendszer telepítése. A Catalyst szabadon letölthető a **CPAN**-ról, ahol rengeteg Perl nyelven írt, szabadon felhasználható modul található. A CPAN a Comprehensive Perl Archive Network rövidítése, amely a Perl közösség által feltöltött programok és dokumentációk összessége, kereshető formában. Meglátogattam a böngészővel a http://search.cpan.org/ webhelyet, és rákerestem a „Catalyst" modulra. Rögtön az első találatra kattintva bejött a keretrendszer oldala, ahonnan le is tölthettem azt tömörített formában. Nyitottam egy konzolt és megkerestem a letöltött állományt. A Midnight Commander segítségével kicsomagoltam a keretrendszer állományait a `/tmp` könyvtárba, majd kiléptem az mc-ből és rendszergazdává váltam. Ezután megkerestem a Catalyst könyvtárban a `Makefile.PL` állományt és elindítottam a `perl Makefile.PL` paranccsal. A program rákérdezett, hogy kézzel szeretném-e beállítani a keretrendszert, vagy inkább rábízom-e a beállítást. Úgy döntöttem, hogy kézzel szeretném beállítani, tehát Entert ütöttem, lévén az „igen" válasz volt az alapértelmezés. Ezután még néhány kérdésnél elfogadtam az alapértelmezett választ, a terminál karakterkódolásánál viszont nemet mondtam az ISO-8859-1 (Latin 1) kódolásra és ezzel az UTF-8 kódolást választottam. Ez fontos lépés volt a magyar mellékjeles karakterek (különösen az „ő" és „ű" betűk) helyes megjelenése érdekében. Még jó néhány kérdés következett, de ezeknél is megfelelő volt az alapértelmezett válasz. Majd ki kellett választanom a megfelelő kontinenst a megfelelő CPAN

tükörkiszolgáló miatt. Természetesen Európát választottam (4-es válasz). Magyarország nem szerepelt az ezután megjelenő listában, így kiválasztottam Ausztriát (1-es válasz), lévén az földrajzilag közel van hazánkhoz. Két tükörkiszolgáló is elérhető volt, mindkettőt kiválasztottam. Kaptam néhány figyelmeztetést függőségek nem teljesüléséről, valamint ajánlatot további két Catalyst kiegészítő telepítésére, amelyek telepítésére később került sor. Most azonban még a keretrendszer telepítésénél jártam. Kiadtam sorban egymás után az ilyenkor szokásos `make`, `make test` és `make install` parancsokat. Az első parancs elkészíti a kész keretrendszert, a második teszteli azt, a harmadik pedig feltelepíti, hogy minden Perl program rendelkezésére állhasson a Catalyst keretrendszer. E parancsok lefutása igénybe vett egy kis időt. A Catalyt sikeres telepítést követően még feltelepítettem a korábban említett kiegészítőket a

```
perl –MCPAN –e 'install Catalyst::Devel'
```

illetve a

```
perl –MCPAN –e 'install Task::Catalyst'
```

parancsok segítségével. Az első kérte további 7 majd újabb 1, később pedig 2 kötelező modul telepítését. Ezek mindegyikét fel is telepítettem, amely meglehetősen sok ideig tartott (mert ez minden egyes modulra egy-egy `make`, `make test` és `make install` parancsot jelentett, a hálózatról való letöltésen kívül). A második parancs hamar lefutott, szemlátomást már nem telepített új modulokat, viszont kiírta a képernyőre, hogy a **Catalyst::Devel** modul naprakész. Mivel a Template Toolkit sablonrendszert kívántam használni a HTML oldalak előállításához, feltelepítettem még a **Catalyst::Helper::View::TT** modult is a `perl –MCPAN –e 'install Catalyst::Helper::View::TT'` parancs kiadásával. A Template Toolkit segítségével könnyedén hozhatunk létre HTML sablonokat és generálhatjuk le belőlük a konkrét HTML oldalt.

Az autentikáció (azaz a bejelentkezés) támogatásához a

```
perl –MCPAN –e \
'install Catalyst::Authentication::Credential::Authen::Simple'
```

paranccsal feltelepítettem az ehhez szükséges modulokat (a telepíteni kívánt modul igényelte még a `Catalyst::Plugin::Authentication`, az `Authen::Simple`, a `Class::Inspector`, és a `Catalyst::Plugin::Session` modulok telepítését is, amelyet végre is hajtott). Az `Authen::Simple::Passwd` modulra volt még szükség, amellyel bejelentkeztethetőek a Linux felhasználók. Ezt is feltelepítettem a szokásos módon.

Az autentikációhoz kapcsolódik a session-kezelés, ehhez pedig először a következő modulokat telepítettem fel (az első tartalmazza az utolsó kettőt):

```
Catalyst::Plugin::Session
Catalyst::Plugin::Session::State
Catalyst::Plugin::Session::Store
```

Ezek után külön fel kellett telepítenem a sessionazonosítót sütiben (cookie) tároló `Catalyst::Plugin::Session::State::Cookie` modult, illetve a session szerveroldali tárolásához a `Catalyst::Plugin::Session::Store::FastMmap` modult (ez utóbbihoz szükséges a `Cache::FastMmap` modul telepítése is).

A bejelentkezéskor küldött jelszavakat titkosítani kell, ezért telepítenem kellett az SSL (Secure Socket Layer) támogatást nyújtó `Catalyst::Plugin::RequireSSL` modult is.

Most pedig az SVN kiszolgáló beállítása következik. Rendszergazdaként munkálkodtam. Létrehoztam egy *svn* rendszercsoportot és egy *svn* rendszerfiókot a `groupadd -r svn` és a `useradd -r -g svn –m -d /srv/svn -s /bin/false svn` parancsokkal, mert az SVN kiszolgáló működéséhez ezen felhasználó és csoport megléte szükséges. Ezután engedélyeztem a **dav** és a **dav_svn** modulokat az Apache2 webkiszolgálóban:

```
a2enmod dav
a2enmod dav_svn
```

A következő lépés a konfigurációs állomány szerkesztése volt:

```
vi /etc/apache2/conf.d/subversion.conf
```

A leglényegesebb sorok a konfigurációs állományban:

```
<Location /repos/weblab>
   DAV svn
   SVNPath /srv/svn/repos/weblab
   # Limit write permission to list of valid users.
   <LimitExcept GET PROPFIND OPTIONS REPORT>
      # Require SSL connection for password protection.
      # SSLRequireSSL
      AuthType Basic
      AuthName "Authorization Realm"
      AuthUserFile /srv/svn/user_access/svn_passwd
      Require valid-user
   </LimitExcept>
</Location>
```

A WebLab projekthez tartozó tároló beállításai láthatók fentebb. A kettőskereszttel kezdődő sorok megjegyzések. Megadtam a tároló helyét, a jelszóállományt az állományrendszerben és autentikációhoz kötöttem az SVN tároló elérését. Ezt követte az SVN repository (tároló) létrehozása az `mkdir -p /srv/svn/repos` paranccsal. Majd az SVN tároló eléréséhez szükséges jelszóállomány létrehozása, tulajdonosának, csoportjának, elérési jogainak beállítása következett:

```
mkdir /srv/svn/user_access
touch /srv/svn/user_access/svn_passwd
chown root:www /srv/svn/user_access/svn_passwd
chmod 640 /srv/svn/user_access/svn_passwd
```

Hozzáadtam a jelszóállományhoz a *hlab* nevű felhasználót (a parancs bekérte a jelszót):

```
htpasswd2 /srv/svn/user_access/svn_passwd hlab
```

Újraindítottam az `rcapache2 restart` paranccsal az Apache2 webkiszolgálót az új beállítások alkalmazása érdekében. Ezután a projekt tárolójának létrehozása, majd a benne

lévő `dav`, `db` és `locks` könyvtárak tulajdonosának és csoportjának rekurzív beállítása volt soron:

```
svnadmin create /srv/svn/repos/weblab
chown -R wwwrun:www /srv/svn/repos/weblab/{dav,db,locks}
```

Ezzel az SVN tároló elkészült és böngészőből elérhető a http://localhost/repos/weblab címen.

2. Tervezés

Ebben a fejezetben a tervezéssel kapcsolatos leírások, ábrák kaptak helyet. Megfogalmazzuk a kifejlesztendő szoftverrel szemben támasztott követelményeket, definiáljuk a fogalmakat és folyamatokat.

2.1. Áttekintés

I. Általános leírás

A WebLab rendszer (a továbbiakban: Rendszer) a már létező HLAB környezetre épül. Feladata webes adminisztrátori felületet biztosítani a laborkörnyezetek időzítéséhez. A Rendszernek kezelnie kell a HLAB környezet konfigurációs állományait, illeszkednie kell a meglévő környezet sajátosságaihoz. A Rendszer felületet biztosít a Cisco képzésben résztvevő oktatók számára.

II. Általános követelmények

1. A Rendszer felépítésének modulárisnak kell lennie, amely azt jelenti, hogy bármikor bővíthető kell legyen a HLAB környezet fejlődési ütemének megfelelően, az informatikai rendszer minimálisnál nagyobb mértékű módosítása nélkül. Az újabb és újabb feladatokat ellátó modulok telepítésének egyszerűnek és gyorsnak kell lennie, más modulokat nem zavarhatnak meg működésükben.
2. Lényeges a Rendszer felületeinek szép kinézete; és a szemet zavaró színek, zavaró funkcióelrendezések kerülése.
3. A Rendszer minden művelet eredményéről tájékoztatja a Rendszer felhasználóját. Hiba esetén értesítést ad a probléma okáról.
4. A Rendszer által biztosított minden módosítási lehetőségnek konzisztensnek kell lennie. A felületeken történő törlések/módosítások nem eredményezhetnek egymásnak ellentmondó bejegyzéseket a konfigurációs állományokban. Mivel egymással párhuzamosan több felhasználó is használhatja a rendszert, biztosítani kell a megfelelő konkurenciakezelést.

5. Amennyiben az üzembe helyezett Rendszer utólagos módosítása szükséges, vagy profilbővítés illetve változás miatt végrehajtott frissítéskor a Rendszernek biztosítania kell, hogy a korábban elvégzett módosítások ne vesszenek el, garantálnia kell a zavarmentes működést.
6. A Rendszernek a laborkörnyezetek szabályos működéséhez – a felhasználó esetleges figyelmetlenségének okán – a bekövetkezett mulasztásokra egy bizonyos szintig figyelmeztesse a Rendszer felhasználóját és a szabálytalan műveletet ne engedje végrehajtani. Részletes leírását lásd a „Szakterületi folyamatok” című alfejezetben.
7. A Rendszernek a HLAB környezet szakterületi fogalmait kell használnia, melyektől nem térhet el. Ezek felsorolása a „Fogalomszótár” című alfejezetben található.

III. Rendszerkövetelmények

Operációs rendszer: openSUSE 10.1
Programozási nyelv: Perl, C, JavaScript
Célhardver: A rendszer igényeihez fog igazodni. A rendszer tervezésekor nem kell figyelembe venni.
Várható felhasználók száma: 5 fő

2.2. Fogalomszótár

Ebben az alfejezetben a különböző fogalmak meghatározása található.

Felhasználó: Azon személyek (a HLAB rendszeradminisztrátor, illetve az oktatók), akik regisztrált felhasználóként szerepelnek a Rendszerben. Minden felhasználó rendelkezik a következő adatokkal:

1. Felhasználónév
2. Jelszó
3. Teljes név

HLAB adminisztrátor: Speciális felhasználó, aki felügyeli a HLAB rendszer működését.

Laboreszköz: Távolról elérendő forgalomirányító (ROUTER) vagy kapcsoló (SWITCH).

Laborkörnyezet: A felhasználó által megtekinthető, laboreszközökből álló környezet.

Eszközfoglalás (foglalás): Az eszközfoglalás a felhasználó és az adott laborkörnyezet kapcsolatának nyilvántartását jelenti.

Eszközfoglalás állapotok:

1. megtekintés alatt
2. megtekintve, engedélyezés alatt
3. engedélyezve, lefoglalva
4. felszabadítva

2.3. Szakterületi folyamatok

Ebben a szakaszban a különböző folyamatok (tevékenységek) megfogalmazása szerepel.

Bejelentkezés: A Rendszerben regisztrált felhasználók számára felhasználónevük és jelszavuk megadásával elérhetővé válik az eszközfoglalás lehetősége.

Laborkörnyezet megtekintése: A bejelentkezett felhasználó megtekintheti a számára engedélyezett laborkörnyezetek listáját.

Eszközfoglalás: A laborkörnyezet megtekintését követően a kiválasztott laborkörnyezeteket foglalásra lehet megjelölni. Az eszközfoglaláskor a Rendszer regisztrálja a foglalás adatait.

Foglalási életciklus: A foglalás életciklusa a megtekintéstől a felszabadításig tart. A Rendszernek az életciklus minden részét nyomon kell követnie. Ez a foglalás állapotainak nyilvántartását jelenti.

3. Fejlesztés

Ebben a fejezetben először megnézem az MVC tervezési minta elméleti hátterét, majd rátérek az MVC tervezési mintán felépített Catalyst keretrendszer rövid ismertetésére. Ezek után kerül sor a tényleges fejlesztési lépések bemutatására.

3.1. A Model-View-Controller (MVC) tervezési minta

Kezdjük azzal, hogy mi is az a tervezési minta (angolul **design pattern**). A tervezési minták objektumközpontú megoldásokat nyújtanak több számtalan létező problémára, amelyekkel más fejlesztők életük során már többször is szembekerültek. Tulajdonképp arról van szó, hogy a saját tapasztalataik alapján készítettek az adott feladat megoldására egy modellt, amelyek persze általában alapos tesztelésen esnek át, s ezeket követve sok fejfájástól kímélheti meg magát az ember. Ez a feladat-megoldási modell általában kiterjeszthető és rugalmasan testre szabható. Létezik már jó néhány minta, melyek kategória szerinti besorolás alapján lehetnek létrehozási minták, szerkezeti minták, viselkedési minták, és így tovább.

Az MVC tervezési minta szerkezeti minta. Elég régen megszületett már a gondolata (1979), azóta számtalan teszten is átment, sok területen alkalmazták már sikerrel, tehát érdemes időt szánni rá. Arról szól tulajdonképpen, hogy változtatni szeretnénk az adatkezelésen, újraszervezni az adatokat és nem szeretnénk, hogy mindez érintse a felhasználói felületet. Az adathozzáférés és az úgynevezett üzleti logika elválik az adat prezentációjától, melyet egy köztes komponens bevezetésével érünk el, ezt hívjuk Controller-nek. Nézzük meg egyenként, hogy mi milyen szerepet játszik az MVC-ben:

Model (modell): Ebben a részben 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. Amennyiben az alkalmazásunk valamilyen adatbázisból veszi az adatokat (az esetek többségében erről van szó) és ezt változtatjuk szerkezetileg, vagy netán kicseréljük (pl. Oracle helyett legyen MS-SQL), akkor jobb esetben csak ezt a réteget kell módosítanunk, s ami még fontosabb, nem vonja magával a felhasználói felület változtatásának szükségét.

View (nézet): A megjelenítéssel kapcsolatos osztályokat írja le, s itt mutatkozik meg az MVC igazi ereje, vagyis szeparálódik a megjelenítés az adatoktól. A View dolga a Model teljes tartalmának vagy egy részének megjelenítése. A legfontosabb alapszabály az, hogy az adatokat csak a Model osztályainak példányain keresztül érhetjük el. Ezt az elvet a helyes tervezés érdekében követni kell. Ha betartjuk ezt az elvet, akkor az eredményt bármilyen felületen keresztül – legyen az böngésző, mobil eszköz – bármilyen tartalmat előállíthatunk a modell alapján. A feladat mindössze annyi, hogy egy-egy View-t készítünk mindhez, melyek ugyanazzal a modellel fognak dolgozni.

Controller (vezérlő): Itt a lényege az egésznek. Rendben, van a Model és vannak View-k, de ezeket össze kellene kötni valahogy. Ezért felel a Controller. Az eseményeket – mint például kattintás egy gombra, billentyűleütés – átfordítja a modell által végrehajtandó akcióra, illetve az akció lefutásának eredményétől függően megjeleníti a megfelelő View-t.

Röviden ennyit az MVC-ről. Rengeteg előnye van a dolognak, mint például 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, amelyek ezt az elhatárolódást szavatolják. Ez nyilván sokkal több időt igényel, de ez a végén busásan megtérül, arról nem is beszélve, hogy mindennek meglesz a helye, nem kell elveszni a nagy összevisszaság áradatában.

3.2. A Catalyst keretrendszer

A Catalyst egy webalkalmazás keretrendszer (**web framework**), amely egyszerre különösen rugalmas ugyanakkor különösen egyszerű is. Hasonló a **Ruby on Rails**-hez, a Java nyelvhez használható **Spring**-hez; illetve a **Maypole**-hoz, amelyen eredetileg alapszik. Legfontosabb tervezési filozófiája, hogy egyszerű hozzáférést biztosítson minden olyan eszközhöz, amely csak szükséges lehet a webalkalmazások fejlesztéséhez, kevés korlátozással. Így megengedi azt is, hogy különböző utakon érjük el a célt. Más webalkalmazás keretrendszerek kezdetben egyszerűbbeknek tűnhetnek, amelyet azonban úgy érnek el, hogy a programozót egyetlen eszközkészlet használatára kényszerítik. A Catalyst nehézségét az jelenti, hogy kicsit többet kell gondolkodni a használatához, pontosan a rugalmassága miatt. Ezt azonban tekintsük

inkább előnynek. Például ez vezet ahhoz, hogy a Catalyst sokkal jobban illik a rendszerintegrációs feladatokhoz, mint más webalkalmazás keretrendszerek.

A Catalyst a fentebb tárgyalt Model-View-Controller (MVC) tervezési mintát követi, amely által lehetővé teszi a tartalom, a megjelenítés, és a folyamatvezérlés külön modulokba szervezését. Ez az elkülönítés teszi lehetővé, hogy ha bizonyos dolgot változtatunk a kódban, akkor a más dolgokért felelő kódra nincs kihatással ez a változtatás. A Catalyst a már létező olyan Perl modulok újrafelhasználását ajánlja, amelyek jól kezelik a webalkalmazásoknál jelentkező szokásos feladatokat.

Lássuk, hogyan fedi le a Model, a View és a Controller ezeket a feladatokat! Említésre kerül példaképpen néhány olyan ismertebb Perl modul is, amely alkalmas lehet az adott réteg feladatának ellátására.

Model: Tartalom (adatok) elérése és módosítása. Történhet például a DBIx::Class, Class::DBI, Xapian, Net::LDAP Perl modulok segítségével.

View: A tartalom megjelenítése a felhasználó számára. Erre használható például a Template Toolkit, a Mason, és a HTML::Template.

Controller: A teljes kérés-feldolgozási fázis, a paraméterek ellenőrzése, az akciók továbbítása, folyamatvezérlés. Mindezeket a Catalyst maga végzi.

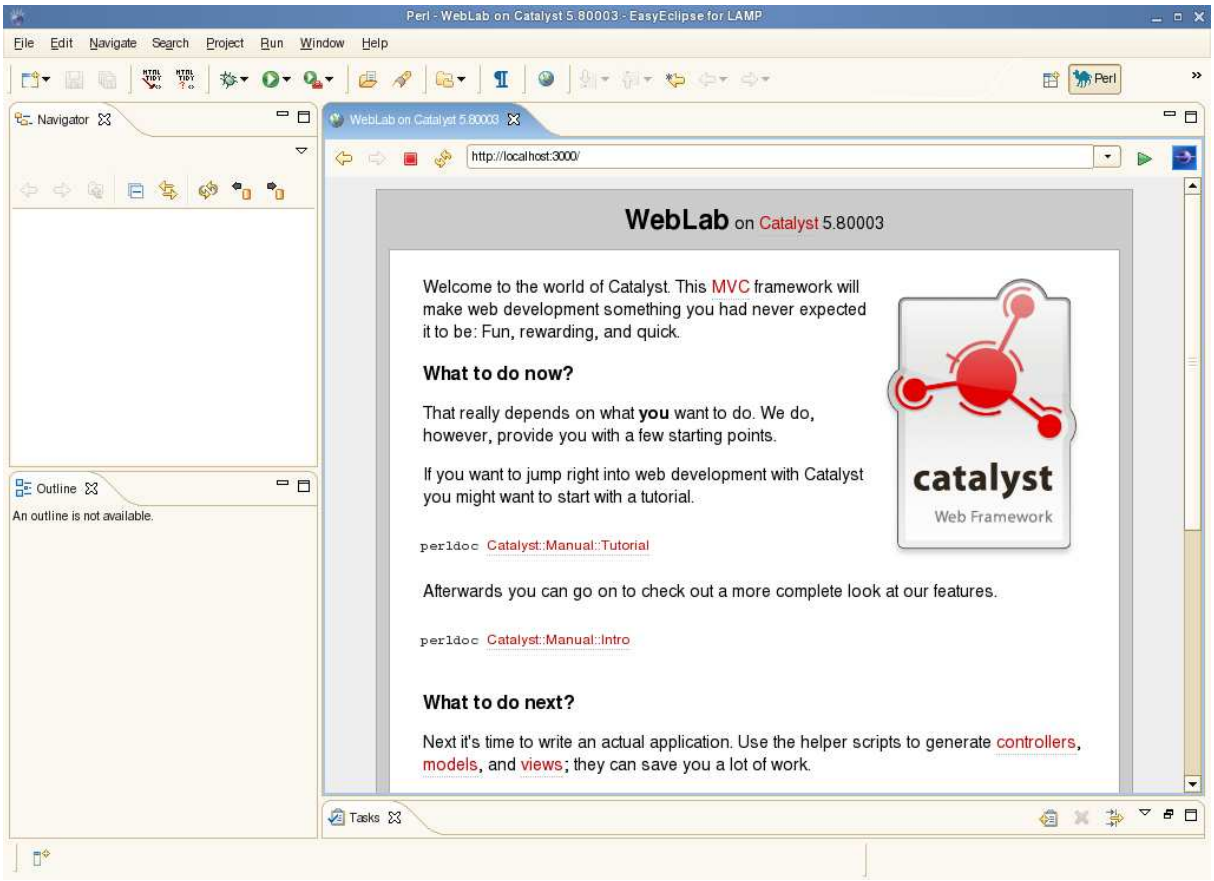
A Catalyst nagy előnye, hogy rendelkezik konzolos eszközökkel is, amelyek segítségével rengeteg dolgot elvégezhetünk, megtakarítva magunknak ezzel rengeteg fáradságos és unalmas munkát.

3.3. Kezdeti lépések

Először is elindítottam az EasyEclipse fejlesztői környezetet, majd nyitottam egy konzolt, és elnavigáltam az EasyEclipse workspace könyvtárába, hiszen itt akartam létrehozatni a Catalyst-tal a webalkalmazás „csontvázát”. A webalkalmazás neve WebLab, így a létrehozása a következő Catalyst paranccsal történt:

```
catalyst.pl WebLab
```

E parancs létrehozta a webalkalmazáshoz tartozó könyvtárszerkezetet és állományokat, tételesen tájékoztatást is adva arról, hogy milyen új könyvtárak és állományok jöttek létre. Kaptam arról is tájékoztatást, hogy lépjek be az alkalmazás könyvtárába, és futtassam le a `perl Makefile.PL` parancsot, hogy meggyőződhessem arról, hogy a létrehozás valóban teljesen sikeres volt-e. Ezt is tettem. Ezután pedig a webalkalmazás `script` könyvtárában található `weblab_server.pl` elindításával tesztelni is tudtam a frissen létrehozott alkalmazást. Ez a program nem más, mint a Catalyst keretrendszer saját webkiszolgáló programja. Előnye, hogy nyomkövetési információkat is szolgáltat: mutatja, ha kérés érkezik hozzá, a kérések részleteivel; mutatja a lefutott akciókat, és így tovább. Alapértelmezetten a 3000-es TCP porton várakozik, tehát a böngészőben (az EasyEclipse tartalmaz beépítettet) a http://localhost:3000 címen érhető el az alkalmazás. Így nézett ki az EasyEclipse beépített böngészőjével:



5. ábra

A kezdőoldal különböző segítségeket ajánlott fel: Ha azonnal bele kívánjuk vetni magunkat a Catalyst-os webfejlesztésbe, akkor egy tutorial-ra lesz leginkább szükségünk, amelynek

megnyitásához adjuk ki `a perldoc Catalyst::Manual::Tutorial` parancsot (vagy egyszerűen kattintsunk rá az oldalon és betöltődik a tutorial CPAN-on található online változata). A későbbiekben a teljesebb kép érdekében pedig a kézikönyvet ajánlotta, amely elérhető a `perldoc Catalyst::Manual::Intro` parancs segítségével, vagy a hivatkozásra kattintással. Kicsit lentebb pedig arról írt, hogyan is írhatjuk meg tényleges alkalmazásunkat, azaz, hogyan hozhatjuk létre a vezérlőket, modelleket és nézeteket. Ezt megkönnyítendő rendelkezésünkre áll a webalkalmazás `script` könyvtárban a `weblab_create.pl` helper script (Catalyst segédprogram). A segédprogram használatáról a `-help` kapcsoló ad tájékoztatást, példákkal. A kezdőoldal felhívta a figyelmemet arra is, hogy érdemes felkeresni a CPAN-t a Catalyst beépülő modulok után kutatva, amelyek igen hasznosak lehetnek, hiszen ezek már elkészített elemek, csak használni kell őket (nyilván előtte kicsit meg kell vizsgálni őket, hogy mire és hogyan is használhatóak). Ha pedig segítségre lenne szükségünk, akkor az igen aktív Catalyst közösséggel több helyen is felvehetjük a kapcsolatot (Wiki, levelezőlista, IRC).

Az így elkészült alkalmazásvázat feltöltöttem az SVN tárolóba az

```
svn import /home/hlab/workspace/WebLab \
http://localhost/repos/weblab -m "Kezdeti import"
```

paranccsal.

Még egy fontos momentumról kell említést tennem, nevezetesen arról, hogyan is tudom ezt a projektet a létrehozott SVN tárolóból az EasyEclipse-be betölteni, hogy a forrásállományokat kényelmesen szerkeszthessem. Az EasyEclipse elindítása után a „File/New” alatti „Project…” menüpontot választottam ki, a megnyíló ablakban kikerestem az SVN típusú projektet. Azt lenyitva az egyetlen megjelenő „Checkout Projects from SVN” sort választottam ki, majd a „Next >” gombra kattintva a „Create a new repository location” feliratú rádiógombot választottam ki és továbblépve megadtam a http://localhost/repos/weblab URL-t. Itt kiválasztottam a gyökeret és a „Finish” gombbal megkezdtem a projektet letölteni az SVN tárolóból. Még kaptam egy figyelmeztetést, hogy esetleg minden (fejlesztési) ágat letöltök. Ezt tudomásul vettem és kezdetét vette a checkout. Ennek végeztével meg is jelentek az EasyEclipse-ben a Catalyst által eddig létrehozott könyvtárak és állományok.

Az alkalmazást mindig egy-két lépés megtételével bővítettem, így építve fel apránként a teljes rendszert. A lépéseket rendszerint dokumentációolvasás előzte meg, hogy milyen eszközöket is bocsát rendelkezésemre a keretrendszer az adott probléma megoldásához, és azokat hogyan is kell használni. Természetesen végigkövettem a tutorial-ban a szokásos „Hello World" alkalmazás elkészítését, és annak bővítését is. Igyekeztem észrevenni, hogy milyen eszközökkel érhetem el célomat.

3.4. Lépésről lépésre a cél felé

A Catalyst a `lib/WebLab` könyvtár alatti `Controller`, `Model` és `View` könyvtárakban tárolja az MVC architektúra elemeit. A `Controller` könyvtárban lévő `Root.pm` modulban módosítottam az `index` szubrutint, amellyel új kezdőlapot adtam meg az oldalhoz:

```
sub index :Path :Args(0) {
    my ( $self, $c ) = @_;
    $c->response->body("Üdvözli Önt a WebLab!");
}
```

A `$self` változó a példányra mutató referenciát, míg a `$c` változó az úgynevezett kontextus-objektumra mutató referenciát tartalmazza. Ez a későbbiekben nagyon fontos szerepet fog játszani, mert segítségével közvetlenül kapcsolódhatunk a Catalyst keretrendszerhez, és köthetjük össze vele az alkalmazás egyes részeit.

A tesztszerver újraindítását követően a http://localhost:3000/ címen megjelent az „Üdvözli Önt a WebLab!" szöveg. Kis magyarázat ehhez: A http://valahol.hu/ címet a Catalyst keretrendszer a *Projektnév*`::Controller::Root` modulban lévő `index` szubrutinnak feleltteti meg (azaz a `Root` vezérlő `index` akciójáról van szó, a Catalyst terminológiája szerint).

Ezután kicsit továbbfejlesztettem ezt a kezdőlapot úgy, hogy létrehoztam a `script/weblab_create.pl view TT TT` paranccsal egy Template Toolkit nézetet. A `view` kulcsszó jelzi, hogy nézetet kívánok létrehozni, az első `TT` a modul (szokásos) neve, a második `TT` pedig arra utal, hogy a nézet típusa Template Toolkit. Ezzel létrejött a

`lib/WebLab/View/TT.pm` modul. A `root` könyvtárban létrehoztam egy `index.tt` sablonállományt, melynek tartalma a következő volt:

```
<p>
  <b>Üdvözli Önt a WebLab!</b>
</p>
```

Az `index` szubrutint is átírtam, hogy az oldal már a sablonállományból jelenjék meg:

```
sub index :Path :Args(0) {
    my ( $self, $c ) = @_;
    $c->stash->{template} = 'index.tt';
}
```

A `stash` szolgál arra, hogy megoszthassunk információkat az alkalmazás többi része számára. A tesztszerver újraindítását követően a http://localhost:3000/ címen félkövéren szedve jelent meg az „Üdvözli Önt a WebLab!” szöveg.

Mivel csak a regisztrált felhasználók léphetnek majd be, fel kellett készítenem az oldalt az autentikációra. Természetesen a Catalyst ehhez is nyújt támogatást.

A `lib/WebLab.pm` modulban a következő módosítások voltak szükségesek az autentikációhoz és a session-kezeléshez (félkövéren szedve a beszúrt sorok, a pontok pedig a számunkra most nem lényeges sorok kihagyását jelzik):

```
package WebLab;
.
use Catalyst qw/-Debug
                ConfigLoader
                Static::Simple
                Authentication
                Session
                Session::Store::FastMmap
                Session::State::Cookie
               /;
```

```
.
.
__PACKAGE__->config(
    name => 'WebLab',
    'Plugin::Session' => {
        cookie_expires => 0,
        expires => 3600,
        storage => '/tmp/session'
    }
);
```

Kliens oldalon, azaz a böngészőben sütiben (cookie) tároljuk az autentikációhoz szükséges adatokat, ezért a program működéséhez engedélyezni kell a sütiket. A bejelentkeztetés alapértelmezés szerint a Linux lokális felhasználók titkosított jelszavait tároló `/etc/shadow` állomány segítségével történik. Azonban a belépéshez nem elegendő, hogy a felhasználó lokális Linux felhasználó legyen, szükséges az is, hogy benne legyen a WebLab-felhasználók listájában is. E lista egy szöveges állományban kerül tárolásra. Az állomány neve (a teljes elérési úttal együtt) megadható a `weblab.conf` állományban. Például:

```
weblab_users /etc/hlab/weblab.txt
```

Ha nem a Linux rendszer `/etc/shadow` állományát használjuk a WebLab felhasználók autentikálásához, akkor a fenti `weblab_users` beállításra nincs szükségünk, így azt tegyük megjegyzésbe a # jel sor elejére való beszúrásával, vagy pedig töröljük ki a konfigurációs állományból. Az autentikációhoz használni kívánt állomány nevét a teljes elérési útjával együtt a `weblab.conf` állományban adhatjuk meg (a beállítások érvényesítéséhez újra kell indítani a webszervert az `apache2ctl restart` paranccsal). Az egyéni autentikációs állományra a `shadow` csoportnak olvasási joggal kell rendelkeznie.

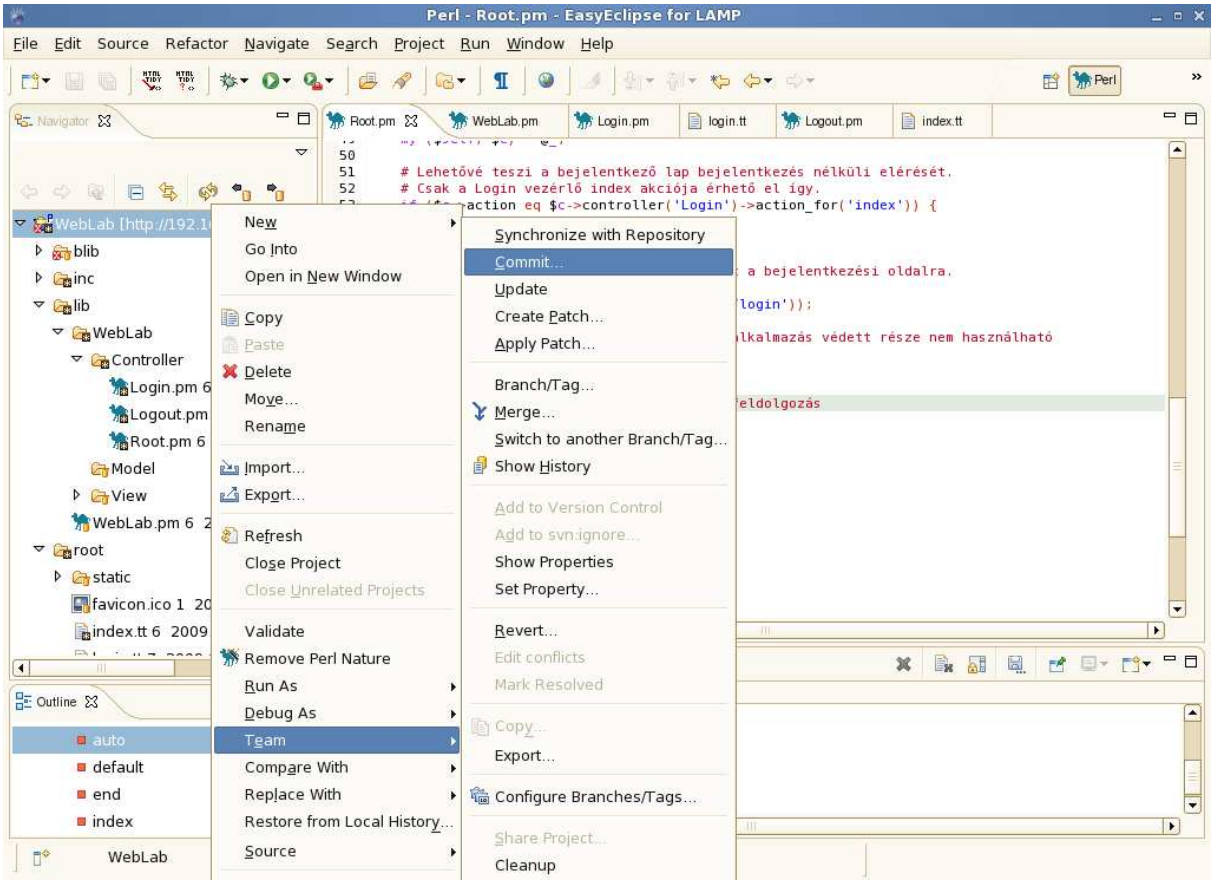
A továbbiakban létrehoztam a be- illetve a kijelentkezés megvalósításához két vezérlőt *Login* és *Logout* néven:

```
script/weblab_create.pl controller Login
```

```
script/weblab_create.pl controller Logout
```

Megoldható lett volna persze akár egyetlen vezérlő két akciójával is (például *Auth* vezérlő *login* és *logout* akciója), a Catalyst keretrendszer rugalmasságának köszönhetően, számomra azonban szimpatikusabb volt ez a két vezérlőt alkalmazó megoldás. A `lib/WebLab/Controller/Login.pm` modulban az `index` szubrutint úgy írtam meg, hogy elvégezze a felhasználónév és jelszó ellenőrzését. Ha a bejelentkezés sikeres, akkor a kezdőoldalra visz át. A `lib/WebLab/Controller/Logout.pm` modul `index` szubrutinja pedig a kijelentkeztetést végzi el. Most már rendelkezett az oldal a be- és kijelentkeztetés funkcionalitásával, de ez még nem volt elegendő: Meg kellett akadályoznom, hogy valaki az oldal olyan részére léphessen (például a megfelelő URL beírásával), amelyhez bejelentkezés szükséges. Azaz el kellett azt érnem, hogy bejelentkezés nélkül csak a bejelentkezéshez szükséges oldal lehessen elérhető. Nagyon fáradtságos lenne minden vezérlő minden akciójához megírni egy ilyen ellenőrzést. Szerencsére a Catalyst erre is ad elegáns megoldást: A *Root* vezérlőben kell létrehozni egy *auto* akciót, amely elvégzi ezt az ellenőrzést. Ez az akció minden más akció előtt fut le, így alkalmas arra, hogy megakadályozza a bejelentkezéshez kötött oldalak elérését. A következő lépés az SSL titkosítás beállítása volt. A `lib/WebLab.pm` modulban a `__PACKAGE__->setup()` szubrutint kiegészítettem a `qw/RequireSSL/` aktuális paraméterrel, ezzel jelezvén a keretrendszernek, hogy használni kívánom az SSL titkosítást. Ezután pedig a *Login* vezérlő *index* akcióját kiegészítettem (az elején) a `$c->require_ssl;` sorral, amely kéri az SSL titkosítás használatát ebben az akcióban. Sajnos a Catalyst beépített teszt webkiszolgálója nem támogatja az SSL titkosítást, így ezt nem tudtam teljesen kipróbálni működés közben. Viszont a nyomkövető információkban megjelent, hogy át akart váltani titkosított módra. Ezzel az autentikáció implementálása lényegében teljesen elkészült, eltekintve a megjelenéstől (A „Előbb a működés, aztán a kinézet” elvet követve). Ez most mindenképpen olyan állomás, amelyet érdemes feltölteni a projekt SVN tárolójába. Ezt az EasyEclipse-ben a projekt főkönyvtárának kijelölésével és azon való jobbkattintással előhozott helyi menü segítségével tehettem meg, a „Team/Commit…” menüpont kiválasztásával, ahogyan az a 6. ábrán látható. Az „Edit the commit comment” mezőbe beírtam az „Elkészült az autentikáció.” szöveget, majd megnyomtam a biztonság kedvéért a „Select All” nyomógombot, hogy

minden új elem bekerüljön a commit-ba. Az „OK” gombra kattintva feltöltöttem a változásokat az SVN tárolóba.



6. ábra

Biztonsági okokból érvénytelen vezérlő megadása esetén átirányítás történik a /-re (ha pedig még nem jelentkezett be a felhasználó, vagy ha már lejárt a session, akkor irányítás a /login-ra). Ezt a viselkedést a `Root.pm` modul `default` és `auto` metódusaiban adhattam meg. A kódrészletek (az `auto` metódusnak csak egy – most lényeges – részét mutatom, pontokkal jelölvén az elhagyott részeket):

```
sub default :Path {

    my ( $self, $c ) = @_;
    $c->response->redirect($c->uri_for('/'));
}

sub auto : Private {
```

```
    my ($self, $c) = @_;
.
.
    if (!$c->user_exists) {
        $c->response->redirect($c->uri_for('/login'));
.
.
}
```

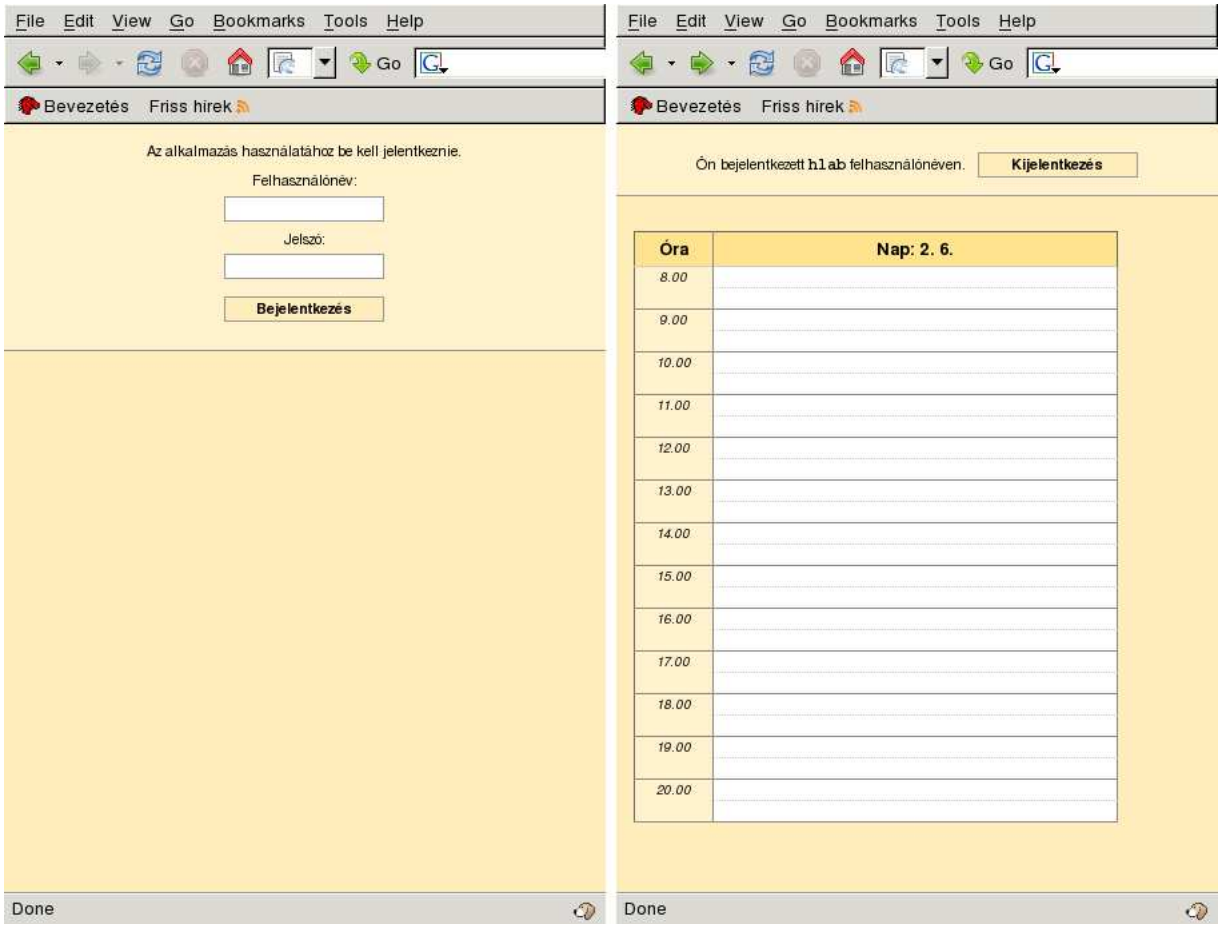
Látható, hogy a `redirect` metódus segítségével tudjuk átirányítani egy másik URL-re (és ezzel egy másik vezérlőre) az alkalmazást.

Bővítsük tovább alkalmazásunkat! A foglalásokhoz egy újabb vezérlőre volt szükségem, ezért létrehoztam egyet *Reserve* néven:

```
script/weblab_create.pl controller Reserve
```

Ehhez a vezérlőhöz készítettem egy újabb Template Toolkit sablonállományt `reserve.tt` néven. Ez határozza meg azt, hogy hogyan is fog kinézni a foglalások kezelését végző felület.

Az `index.tt` sablonállományt bővítettem oly módon, hogy hivatkozzék további két, nevezetesen a `login.tt` és `timetable.tt` sablonra. Az előbbi a bejelentkezéshez szükséges űrlapot tartalmazza (ha még nem jelentkeztünk be), belépés után pedig arról tájékoztat, hogy milyen felhasználónéven léptünk be. A kilépést egy nyomógomb segítségével teszi lehetővé. A második sablon az órarend (azaz a lefoglalt és szabad időintervallumok) megjelenítését szolgálja. Ebben az órarendben lehet majd információkérést, módosítást és törlést végrehajtani. A 7. ábra mutatja, hogyan nézett ki az oldal belépés előtt, illetve belépés után.



7. ábra

E látvány eléréséhez el kellett készíteni a megfelelő vezérlőket, a sablonokat, és a stílusleíró (CSS) állományt. Komolyabb projektekben szükség van designer-re is, akinek nem „csak” annyi a feladata, hogy megálmodja az alkalmazás kinézetét, hanem az is hozzátartozhat a munkájához, hogy a különböző fajtájú és verziójú böngészőkben is megfelelően jelenjen meg az alkalmazás. Ez nagy kihívást jelent, hiszen nem minden böngésző követi teljes mértékben a szabványos megoldásokat. Viszont most be is fejezem az erről való elmélkedést, és visszatérek a projekthez.

Most már rendelkeztem egy üres órarenddel. Ez viszont nem a megszokott formát követi, hiszen csupán egyetlen napot mutat. Szükség volt egy naptárra is, hogy lehessen váltogatni a napokat. A Yahoo! Developer Network-ről letöltöttem a YUI 2 – Yahoo! User Interface Library részét képező Calendar vezérlőt, amelynek segítségével naptárral bővíthettem a webalkalmazást. Ez a naptár JavaScript alapú, a formázást pedig CSS stílusleírással oldja meg. Kissé át kellett szerkesztenem a forráskódot, a CSS állományt szinte teljes egészében,

valamint megfelelő paraméterezéssel kellett meghívnom a naptárt a `timetable.tt` sablonban, hogy olyan viselkedést és kinézetet kapjak, amely illeszkedik az alkalmazáshoz.

A következő lépés a program azon képességének hozzáadása volt, hogy meg tudja jeleníteni a már megtörtént foglalásokat a naptárral kiegészített órarendben. E foglalások meghatározott szerkezetű szöveges állományban kerülnek tárolásra. Az adatsorok a következő „oszlopokból” állnak: hónap, nap, kezdés, befejezés, felhasználónév, laborkörnyezet betűjele. A hónap és a nap(ok) természetes számokkal jelöltek, a kezdet és a befejezés óra:perc formában szerepel. Több nap is megadható egy sorban, vesszővel elválasztva. Természetesen ekkor mindezen napokhoz ugyanaz a kezdés és befejezés tartozik. Például a hetente ismétlődő laborgyakorlatok esetén pontosan ez a helyzet. Lássunk tehát egy példát a foglalásokat tartalmazó állományra:

```
#
# hlab access timing configuration file
#
#============================================================
# File entry format:
#month    day          start - stop      username  A/B/C/D/E/F
#                     hh:mm   hh:mm
#============================================================
02      8,22,23,26    12:15 - 23:46     testuser  A
02      7,28          14:00 - 14:25     testuser  B
```

A Perl itt ismét remekel, hiszen a szövegfeldolgozásban szinte verhetetlen. A sorok adatainak kinyerésére alkalmas reguláris kifejezés:

```
/^\s*(\d{2})\s+(\d{1,2}(?:,\d{1,2})*)\s+(\d{2}:\d{2}) - ↵
(\d{2}:\d{2})\s+(\S+)\s+([A-F])/
```

Némi magyarázatra azért szorulhat a fentebbi kifejezés. Valójában ez a reguláris kifejezés egy sorba írandó (balra mutató nyíl jelzi azt, hogy a következő sor az előző sorhoz tartozik). A kezdő `/` (perjel) a minta kezdetét jelzi. A `^` (kalap) előírja, hogy a sor elejétől kezdődjék az illesztés. A `\s*` jelzi, hogy a sor elején lehet tetszőleges (akár nulla) számú elválasztó

karakter (itt például szóköz vagy tabulátor). Ezzel kizárjuk a `#`-tel (kettős kereszttel) kezdődő sorokat a találatból, lévén, hogy a `#` nem szóköz jellegű karakter. Ez pontosan megfelelő viselkedés számunkra, hiszen ezek megjegyzésnek minősülő sorok, amelyeket nem kell figyelembe venni a foglalások kezelésénél. A `\d{2}` pontosan két decimális számjegyre illeszkedik, amelyet a `\s+` miatt legalább egy elválasztó karakternek kell követnie. Ezután a `\d{1,2}` szerint 1 vagy 2 decimális számjegy következik, amelyet követhet még tetszőleges számú (akár nulla is), vesszőkkel elválasztott 1 vagy 2 decimális számjegy. Majd újra legalább egy elválasztó karakternek kell szerepelnie, amelyet két decimális számjegy, egy `:` (kettőspont), és ismét két decimális számjegy követ. Egy szóköz után egy `-` (mínusz) jel jön, amelyet egy szóköz kísér. Aztán ismét két decimális számjegy, egy kettőspont, két decimális számjegy következik, amely után legalább egy elválasztó karakter szerepel. A `\S+` a nem elválasztó karakterek sorozatára illeszkedik (itt most az angol ABC kis- és nagybetűit és a decimális számjegyeket keressük, mivel ilyen karakterek fordulhatnak elő a felhasználónévben). A már ismerős `\s+` (legalább egy elválasztó karakter) után a `[A-F]` következik a sorban. Ennek jelentése: A és F között az angol ABC bármely nagybetűje.

Még valamire felfigyelhetett az Olvasó, és ez a kerek zárójelek használata. Ez a minta nagyon fontos része, mert így kimenthetők az illeszkedő részek. A Perl mindig menti a kerek zárójelek között lévő találatokat a `$1`, `$2`, … változókba. Ha viszont nem szeretnénk menteni a kerek zárójeleken belüli találatokat, akkor a nyitó zárójelet közvetlenül a `?:` (kérdőjel-kettőspont) kövesse. Itt két esetben voltam kénytelen kerek zárójelet használni, hogy a `*`-gal (csillaggal) kifejezett ismétlés több, a mintában megadott jelre legyen érvényes. Viszont menteni nem kívántam ezeket a találatokat. A Perl a nyitó kerek zárójelektől sorszámozza a mentéseket. Most összesen 6 darab találatot mentünk, amelyek rendre a `$1`, `$2`, …, `$6` nevű változókba kerülnek.

A foglalásokat tároló szöveges állományt tekinthetjük egy egyszerű adatbázisnak, és ily módon logikailag a modell rétegbe sorolható. A modellben kerülnek implementálásra az adatok olvasását és írását lehetővé tévő metódusok.

Létrehoztam az állományalapú modellt a helper szkript segítségével:

```
script/weblab_create.pl model File
```

A modell a beszédes `File` nevet kapta. A következő lépésben a projekt főkönyvtárában lévő `weblab.conf` nevű állományába beszúrtam a következő sort:

```
hlab_file /etc/hlab/hlab.txt
```

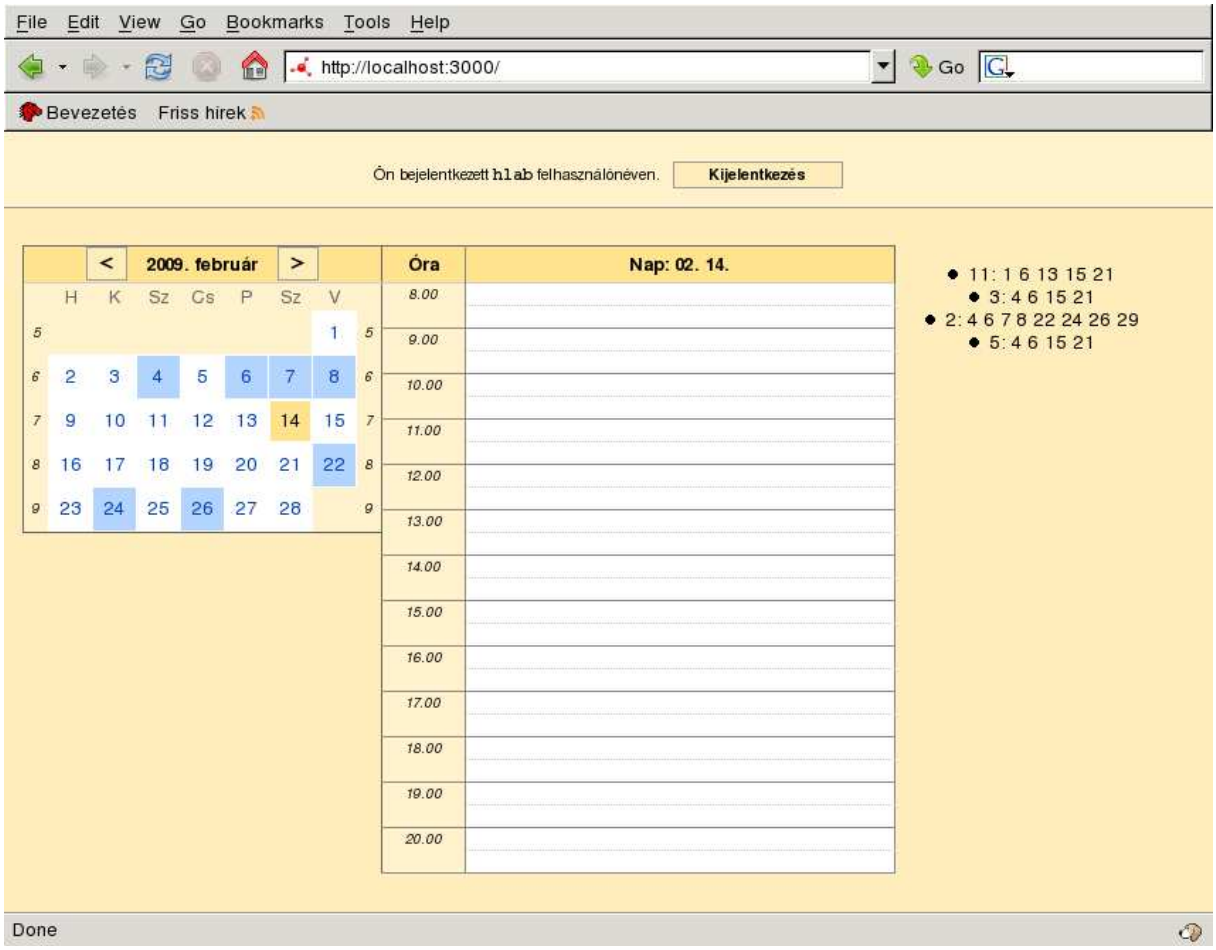
Ezzel megadtam a foglalásokat tároló szöveges állomány nevét a teljes elérési útjával együtt. A modellben a `WebLab->config->{hlab_file}` hívással juthatunk majd hozzá ehhez a névhez. Látható, hogy milyen hasznos, ha a különféle paramétereket a konfigurációs állományba vesszük fel, mert ezután a projekt bármely részében lekérdezhetjük azokat.

Ahogy említettem, a modellbe kerülnek az olvasást és írást megvalósító metódusok. Igen ám, de hogyan is lenne érdemes leképezni az adatokat a memóriába? A kereshetőség és az egyszerűség hogyan ötvözhető ebben a leképezésben? Mely adat(ok) szerint lenne érdemes indexelni? Ezekre a kérdésekre kellett választ találnom.

Végül a következő megoldást választottam: Egy hash-t használok, amelynek szöveges kulcsai ilyen formában állnak elő: `'hónap;kó:kp;bó:bp;felhnév;labkörny'`. A kulcsok pedig a napokat tartalmazó numerikusan rendezett tömbre hivatkoznak. A `kó` a kezdés órájára, a `kp` a kezdés percére, a `bó` a befejezés órájára, a `bp` a befejezés percére, a `felhnév` a felhasználónévre utal. A `labkörny` a laborkörnyezetet jelzi. Ez konkrétan például így jelenhet meg: A vagy B, vagy C, egészen F-ig. Azért döntöttem ilyen kulcs használata mellett, mert így összefésülhetők a több sorban lévő, de valójában összevonható foglalási adatok. A `File` modell `get_reservs` metódusa állítja elő a foglalások adatait tároló hash-t. Ebben a metódusban kerül felhasználásra korábban bemutatott reguláris kifejezés az adatok állományból való kinyeréséhez. A metódus használ egy `num_sorted_union` nevű segédfüggvényt, amely két tömb numerikusan rendezett unióját állítja elő (Valójában a függvény két tömbre mutató hivatkozást kap paraméterül, és tömbbel tér vissza). E függvény segítségével tudjuk összefésülni az összevonható foglalási adatokhoz tartozó napokat.

Most már a memóriában voltak a foglalási adatok, így ezután a foglalások órarendben való megjelenítése következett. Az eredmény a 8. ábrán látható. A kék hátterű napokon már van legalább egy foglalás. Az órarend jobb oldalán nyomkövetési információk szerepelnek. Soronként az egy hónaphoz tartozó foglalási napok szerepelnek **hónap_sorszáma: nap1**

nap2 … formában. Ezek az adatok már a foglalásokat tároló szöveges állományból származnak.

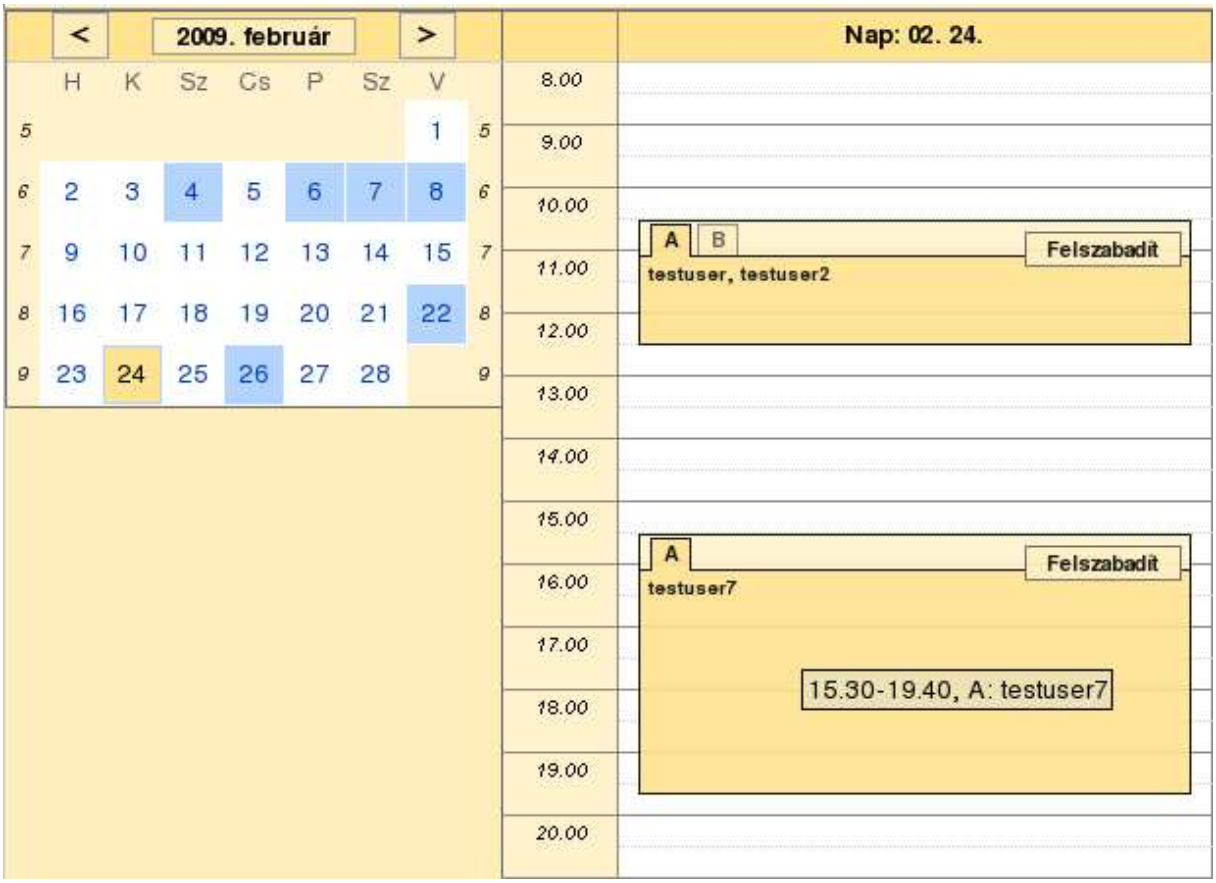


8. ábra

A `File` modell `select_reservs` metódusa számára legfeljebb két paraméter adható meg: a hónap és a nap. Ha csak a hónapot adjuk meg (pontosabban annak sorszámát, például január esetén 1-et), akkor az ahhoz a hónaphoz tartozó összes foglalási adattal tér vissza. Ha megadjuk a napot is, akkor pedig a hónap adott napjához tartozó foglalási adatokat kapjuk meg.

A naptár tehát már mutatott valamit a foglalásokból. Kicsit továbblépve a táblázatban való megjelenítés volt soron. Ehhez külön JavaScript állományt készítettem a `root/static/scripts` könyvtárban `timetable.js` néven. Ebben került definiálásra a `Timetable` nevű osztály, amely három metódussal rendelkezik. Az `addRes` metódus egy foglalási adat táblázatba helyezésére szolgál. A `removeAllRes` az

összes foglalási adatot eltávolítja a táblázatból (erre egy másik nap kiválasztásakor van szükség). A `render` metódus pedig kirajzolja a foglalási adatokat, amelyek fülekkel ellátott téglalapokban jelennek meg a táblázatban, hogy az egy időintervallumhoz, de különböző laborkörnyezetekhez tartozó foglalási adatok egy táblázatban lehessenek megjeleníthetők, kezelhetők. A 9. ábrán megtekinthető egy példa.



9. ábra

A fülek feliratai a laborkörnyezetek betűjeleinek felelnek meg. A felhasználónevek vesszővel kerülnek elválasztásra. Az időintervallumok pedig hozzávetőlegesen meghatározhatók a táblázat baloldalán szereplő időadatok segítségével. Ha elidőzünk az egérrel valamely foglalási adaton, akkor a megjelenő tooltip szövegéből megtudhatjuk a pontos időintervallumot. A tooltip szövege a következő elemekből áll: kezdés ideje (óra és perc), befejezés ideje (szintén óra és perc), laborkörnyezet betűjele, valamint a felhasználónevek vesszővel elválasztott listája. A lapfüles szerkezet kialakításához Patrick Fitzgerald JavaScriptben megírt MIT licensz alatt kiadott kódját használtam fel, amely a lapfüles elrendezést egymásba ágyazott `div` HTML tag-ekből hozza létre. Álljon itt egy rövid példa,

amely két füllel rendelkező lapot hoz létre. A lapfülek címkéje A és B, az A címkéjű lapon két felhasználónév, a B címkéjűn pedig egy felhasználónév szerepel:

```
<div class="tabber">
  <div class="tabbertab">
    <h2>A</h2>
      testuser1, testuser2
  </div>
  <div class="tabbertab">
    <h2>B</h2>
      testuser3
  </div>
</div>
```

A fenti kód rövid magyarázata: A „tabber” CSS osztályba tartozó külső `div` tag fogja közre a „tabbertab” CSS osztályba tartozó `div` tag-eket, amelyek tartalmazzák a lapfül címkéjét valamilyen header HTML tag-ek (itt `h2`-k) között, és ezt követően a lapfül tartalmát.

Annak érdekében, hogy a naptárban egy nap kiválasztása ne eredményezze az egész oldal újratöltését, de mégis frissüljenek a táblázatban a foglalási adatok, az AJAX (Asynchronous JavaScript and XML) használata szükséges. Ekkor a webalkalmazás kis mennyiségű adatot cserél a háttérben, amely megnöveli a teljesítményt. Az adatcserében résztvevő adatok pedig a JSON (JavaScript Object Notation) által meghatározott formátumban utaznak a hálózaton. Ez szöveges formátum, amely ember által is olvasható módon jeleníti meg az egyszerű adatstruktúrákat és asszociatív tömböket.

A JSON-támogatáshoz feltelepítettem a `Catalyst::View::JSON` modult. Látható, hogy a JSON a Catalyst terminológiájában a View-ba tartozik. Ezután létrehoztam a `script/weblab_create.pl view JSON JSON` paranccsal JSON néven egy JSON nézetet. A `WebLab.pm` modulban a megfelelő működéshez a következő konfigurációs elemeket kellett felvennem (félkövéren szedve szerepelnek):

```
__PACKAGE__->config(
.
.
    'View::JSON' => {
          allow_callback  => 0,
          expose_stash    => 'res_on_day'
    },
    'default_view' => 'TT'
.
.
}
```

A callback lehetősége letiltásra került, mivel most nincs szükség szkriptek futtatására, egyszerűen csak adatokat küldünk át. Ennél fontosabb beállítás, hogy milyen nevű kulcso(ka)t figyeljen a JSON a stash-ben. A megadott kulcs(ok) alatt lévő adatokat fogja használni a válaszküldéshez. A stash res_on_key kulcsa alatt található az adatstruktúra, amely egy adott nap összes foglalási adatát tartalmazza. Pontosan erre van szükségünk egy nap kiválasztásánál. Azonban eddig még csak a kiszolgálóoldalról beszéltünk. Szerencsére az ügyféloldalon a YUI naptár könnyedén kiegészíthető JSON támogatással.

Nem esett még szó a `default_view` beállítás magyarázatáról. Ennek értéke `TT`, azaz Template Toolkit. Erre azért van szükség, mert most már nem csak egyetlen View-val rendelkezik az alkalmazás, de továbbra is szeretnénk HTML-oldalakat az ügyféloldalon. Ezeket pedig a Template Toolkit állítja elő, nem a JSON.

Az alkalmazás immár meg tudta jeleníteni a foglalási adatokat. Következő lépés egy olyan űrlap kialakítása volt, amely lehetővé teszi a foglalási adatok bevitelét. Ehhez össze kellett gyűjteni, hogy milyen adatok bevitelére lehet szükség. Ezek a következők voltak: hónap, nap, kezdés, és befejezés időpontja (óra, perc), felhasználónevek, laborkörnyezet betűjele. A létrehozott űrlapon az idő jellegű adatok legördülő listából választhatók ki, a felhasználónevek szövegmezőbe gépelhetők be, a laborkörnyezet betűjele pedig rádiógomb csoportból választható ki; továbbá kérhető a foglalás ismétlődése (az adott hónapban) egy kiválasztónégyzet segítségével. Általános szabályként elmondható, hogy mindig nagyon lényeges a felhasználó által bevitt adatok ellenőrzése, az okozott károk elkerülése érdekében.

A hibás bevitel helytelen működést eredményezhet, és támadásnak is kiteheti az alkalmazást, valamint az alkalmazást futtató kiszolgálót. Webalkalmazások esetén alapvetően két helyen végezhető el ez az ellenőrzés: Kiszolgáló- és/vagy ügyféloldalon. Jelen alkalmazás esetében mind a két helyen történik ellenőrzés. Az ügyféloldali adatbevitel-ellenőrzés előnye az, hogy nem a kiszolgálót terheli, és az eleve helytelen adat ki sem kerül a hálózatra, csökkentve a felesleges hálózati forgalmat. Hátránya viszont, hogy az ügyféloldal felett nem rendelkezünk fennhatósággal, így nem garantált, hogy az ellenőrzés megtörténik-e, illetve hogy úgy történik-e meg, ahogy azt elvárjuk. A kiszolgálóoldali ellenőrzés felett általában teljes ellenőrzésünk lehet, viszont ez a kiszolgálón okoz terhelést.

A bevitt adatok ellenőrzését kliensoldalon JavaScript és CSS segítségével oldottam meg. JavaScript-ben is reguláris kifejezéseket használtam az adatbevitel formai helyességének ellenőrzéséhez (például a felhasználóneveknél). A dátumnál és az időadatoknál értékvizsgálatra is szükség volt (például: létezik-e az adott nap; a kezdés korábban van-e a befejezésnél). A CSS segítségével pedig piros keretet adhattam azoknak a beviteli mezőknek, amelyben hibás bevitel szerepel (a JavaScript kódban átállítom a hibás bevitelt tartalmazó vezérlő CSS-osztályát). A továbbiakban szerepeltetek néhány példát az ellenőrzések közül.

A felhasználónevek formai helyességének ellenőrzésére a következő reguláris kifejezést (mintát) alkalmaztam: `/^[a-z]([a-z]|\d)*$/`

Magyarázat: Olyan felhasználóneveket várunk, amelyekben az angol ABC kisbetűi és decimális számjegyek szerepelhetnek. A felhasználónévnek betűvel kell kezdődnie és legalább egy karakterből kell állnia.

A JavaScript kódban írtam egy az előbbi mintát felhasználó függvényt, amely arra szolgál, hogy egy szöveges HTML űrlapelembe bevitt szöveget feltördel szóközök és vesszők mentén, az eredményt elhelyezi tömbben, amelyet bejár (teljesen vagy részben), a tömbelemeket vizsgálva, hogy eleget tesz-e a mintának (azaz, hogy formailag helyes felhasználónév-e). Ha már egyetlen egy formailag helyes felhasználónevet talál, igaz értékkel tér vissza, különben pedig hamissal. Ez a viselkedés azért engedhető meg, mert a kiszolgálóoldalon is történik érvényességvizsgálat, ahol az érvénytelen felhasználónevek egyszerűen eldobásra kerülnek. A függvény kódja:

```
function has_username(id) {
  var text = document.getElementById(id).value;
  var unames = text.split(/,|\s+/);
  for(var i in unames) {
    if(unames[i].match(/^[a-z]([a-z]|\d)*$/)) {
      return true;
    }
  }
  return false;
}
```

Mivel mindenképpen ki kell választani egy laborkörnyezetet, ennek ellenőrzésére is írtam egy függvényt:

```
function sel_env() {
  var envs = ['enva', 'envb', 'envc', 'envd', 'enve', 'envf'];
  for(var i in envs) {
    if(document.getElementById(envs[i]).checked) {
      return true;
    }
  }
  return false;
}
```

Az időadatoknál létrehozok két JavaScript `Date` objektumot `bd` (begin date) és `ed` (end date) néven, amelyeknél az év, hónap és nap megegyezik, az óra és perc pedig a megfelelő kezdési és befejezési időpontok. Annak ellenőrzése, hogy a kezdés és befejezés időpontja között legalább 1 óra van, a következő módon tehető meg:

```
ed.valueOf() - bd.valueOf() >= 3600000
```

A JavaScript `Date` objektum `valueOf()` metódusa az 1970. január 1. éjfél után eltelt ezredmásodpercek számával tér vissza. Ezért szerepel a fenti kódsorban ez az óriási számérték (1 óra = 3600 másodperc, ezt kell még megszorozni 1000-rel az ezredmásodpercben megadott érték miatt).

Az űrlap adatainak ellenőrzésére a `form` HTML tag `onsubmit` attribútumában megadható egy logikai típusú függvény. Ha ez a függvény igaz értékkel tér vissza (ezzel jelezhető, hogy a bevitel helyes), akkor az űrlap adatai elküldésre kerülnek a kiszolgálóra, további feldolgozásra. Ha pedig hamis értékkel tér vissza, akkor az űrlapadatok nem kerülnek elküldésre. Itt tehát be tudunk avatkozni a folyamatba egy megfelelően megírt függvény segítségével.

Most térjünk át a bevitt adatok szerveroldali ellenőrzésére. A `Reserve` vezérlő `index` akciójában került implementálásra a bevitel ellenőrzésének nagy része. A böngészőtől érkező adatok karaktersorozatokként érkeznek, amelyek első körben trimmelésre kerülnek, azaz a karaktersorozatok elejéről és végéről eltávolításra kerülnek a szóközök. Ehhez a következő reguláris kifejezést használtam: `/^\s*(\S*)\s*$/`

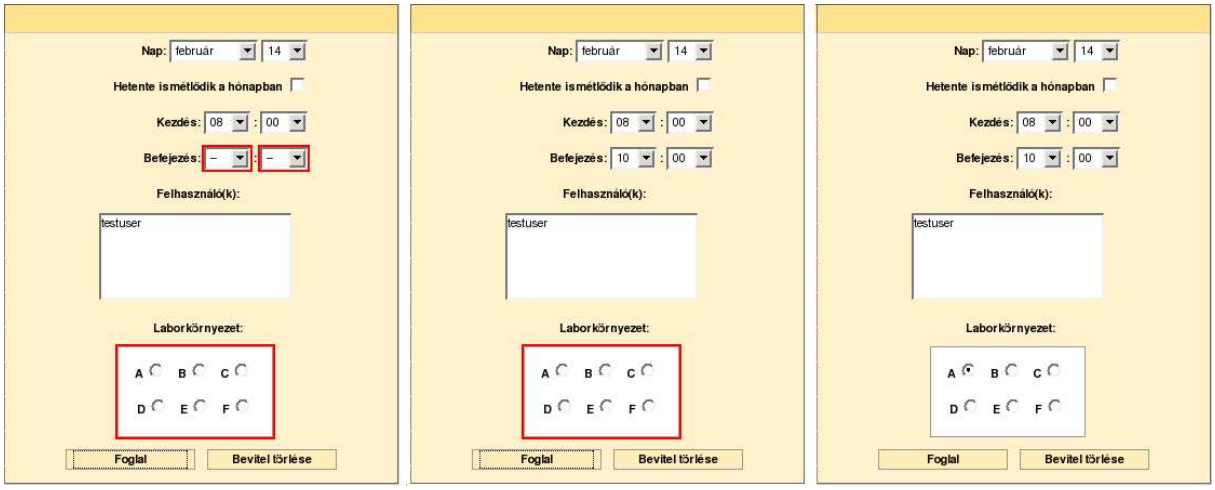
Ez nem általános trimmelő, mert ha a bevitel belsejében is található szóköz, akkor nem ad találatot. Itt most azért alkalmazható mégis, mert a bevitelek a belsejükben nem tartalmazhatnak szóközt.

Ellenőrzésre kerül az üres bevitel, mivel mindenképpen meg kell adni az egyes bevitelekhez valamit, nem lehetnek üresek.

Az időadatoknál inkább csak formai ellenőrzések vannak reguláris kifejezések segítségével. A hónap, nap alaki helyességét a `/^\d{1,2}\.\d{1,2}$/` mintával, míg az óra, perc formai ellenőrzését a `/^\d{1,2}:\d{1,2}$/` mintával végzi el a program.

A felhasználóneveknél leválogatásra kerülnek a formailag helyesek, a program csak ezekkel dolgozik tovább. Az itt használt reguláris kifejezés megegyezik a kliensoldali ellenőrzésnél alkalmazottal.

Következzék néhány képernyőkép az űrlap különböző megjelenéseiről, attól függően, hogy érvényes-e a bevitel vagy sem. A 10. ábrán látható első űrlapon a befejezés ideje és a laborkörnyezet betűjele nincs megadva, a „Foglal” feliratú gomb megnyomása után a „Befejezés” címke mellett szereplő két legördülő lista (befejezés órája, perce) és a „Laborkörnyezet” címke alatti rádiógomb csoport (laborkörnyezet) piros keretet kap. A következő űrlapon már csak a laborkörnyezet betűjele nincs kiválasztva. A harmadik űrlapon minden adat helyesen szerepel.



10. ábra

Ismétlődés kérésekor a napok listáját szerveroldalon készíti el a program. Kódrészlet:

```
if(defined($repeat)) {
  my $max = 29;
  my @days = ();
  my $d = $day;

  if($month == 4 || $month == 6 ||
     $month == 9 || $month == 11) { $max = 30;}
  else { $max = 31; }

  push(@days, $d);

  while($d+7 <= $max) { $d += 7; push(@days, $d); }

  $day = join(',', @days);
}
```

A bevitel ellenőrzése után a foglalások állományban történő rögzítése következett, amely az MVC szerint a modellbe tartozik. A `File` modul `write_reserv` metódusa szolgál a foglalások állományba mentésére. Egészen pontosan ez a metódus meghív egy C nyelven megírt bináris burkoló (wrapper) programot, amely meghív egy Perl nyelven megírt programot. Kicsit bonyolultan hangzik, de erre a jogosultságok miatt van szükség. A

webkiszolgáló ugyanis korlátozott felhasználó nevében fut, amely felhasználó nem rendelkezik írási jogosultsággal a foglalásokat tartalmazó állományhoz. A burkoló programra beállítottam az úgynevezett *setuid* bitet, amely azt eredményezi, hogy a meghívott program a tulajdonos jogaival fut és nem a programot meghívó felhasználó jogaival. A tulajdonos pedig a `hlab` felhasználó, amelynek van joga írni ebbe az állományba. A bináris burkoló programra azért van szükség, mert a Linux/Unix rendszerek nem mindegyike engedélyezi a szkriptnyelven megírt setuid-os programokat. Ennek leginkább biztonsági okai vannak, hiszen ezek a szkriptek egyszerű szöveges állományok, amelyek tartalma nagyon egyszerűen olvasható, és így könnyen kereshetők gyenge pontjaik. A biztonság növelése érdekében a bináris burkoló által hívott Perl program csak a `hlab` felhasználó számára olvasható és futtatható, más felhasználó semmilyen joggal sem rendelkezik felette. A foglalási állomány írásához kizárólagos zárolás szükséges, így megvalósítva a kölcsönös kizárást. Ha nem biztosítanánk a kölcsönös kizárást, akkor a két vagy több egy időben történő írás eredménye kérdéses lenne. Az egy időben történő írási kérelmek pedig azért fordulhatnak elő, mert egyszerre több felhasználó is bejelentkezhet (akár különböző helyekről is), és egymástól függetlenül tevékenykedhet.

A felszabadításhoz egy újabb vezérlőt hoztam létre `Free` néven:

```
script/weblab_create.pl controller Free
```

Az újonnan létrehozott vezérlő index akciójában meghívásra kerül a `File` modul `free_reserv` metódusa, amely a `write_reserv` metódushoz hasonlóan működik, csak itt a bináris burkoló program egy másik Perl programot hív meg. A felszabadításnál a szöveges állományból elhagyja a felszabadítandó foglalásokat. Ezt úgy teszi, hogy olvassa az eredeti állományt, és csak azokat a sorokat (illetve sorrészleteket) írja ki az új állományba, amelyeket nem érinti a felszabadítás. Az új állomány neve a régiből képződik úgy, hogy hozzáadódik a névhez a program folyamatazonosítója (PID), amely a `$$` Perl változóban kerül tárolásra. Végül az eredeti állomány nevéhez hozzáfűzésre kerül a „.bak” karaktersorozat, így képezve biztonsági mentést; valamint az új állomány átnevezésre kerül a régi állomány nevére. Végeredményben előáll a foglalási állomány új változata.

Ezzel végére is értünk a fejlesztés bemutatásának. A következő alfejezetben a telepítési csomag elkészítését mutatom be.

3.5. Telepítési csomag elkészítése, telepítése

A telepítési csomag lényegében egy `weblab-`*`verzioszam`*`.tar.gz` állomány, amely tartalmaz a program telepítéséhez minden olyan elemet, amelyre szükség lehet egy HLAB alapkonfiguráción. Ezek az elemek a következők:

- A program állományai (bináris, szkript, és beállítás állományok),
- szükséges Perl modulok,
- Apache FastCGI modul (RPM-csomag),
- tanúsítvány az SSL kapcsolathoz,
- telepítő szkript.

A program állományai lényegében megegyeznek a forráskóddal, mivel a Perl szkriptek nem kerülnek bináris állományokba fordításra. Egyedül a C nyelven írt forrásprogramok maradnak ki a csomagból.

A szükséges Perl modulok azért kerültek összegyűjtésre, hogy azokat ne a CPAN-ról kelljen telepíteni. Sajnos jó néhány modulnál nincsenek megfelelően megadva a függőségek, ezért az automatikus telepítés nem zökkenőmentes. Előny még, hogy nem függ az alkalmazás futása illetve futtathatósága attól, hogy a rendszerben milyen Perl modulok állnak rendelkezésre. A gyűjtemény hátrányaként viszont megemlíthető, hogy a benne lévő modulok frissítése valószínűleg nem fog megtörténi, így azok nem lesznek naprakészek.

Az Apache FastCGI modult azért mellékeltem, mert alapértelmezés szerint nem része a HLAB alapkonfigurációnak, de az alkalmazás futtatásához szükség van rá.

Készítettem egy példa tanúsítványt az SSL kapcsolathoz. Ezt érdemes lehet lecserélni egy olyan tanúsítványra, amelyet megbízható hitelesítés-szolgáltató bocsátott ki. Mindazonáltal a böngészőben feltűnő figyelmeztetés ellenére engedélyezhető ennek a példa tanúsítványnak a használata.

A telepítő szkript Perl nyelven íródott és feladata a program állományainak megfelelő helyre másolása, az Apache FastCGI moduljának telepítése, valamint az Apache webkiszolgáló program konfigurálása, illetve a megfelelő hozzáférési jogosultságok beállítása.

A telepítő szkriptet csak a `root` felhasználó futtathatja eredményesen, minden más felhasználó nevében indítva hibaüzenetet ír ki a szabványos hibacsatornára és befejezi a futását. Erre azért van szükség, mert egyedül a `root` felhasználó rendelkezik mindazokkal a jogosultságokkal, amelyek a telepítés elvégzéséhez szükségesek.

Alapértelmezés szerint a `/opt/WebLab` könyvtárba kerül telepítésre az alkalmazás. Ez az alapértelmezés megváltoztatható az `install.pl` állomány szerkesztésével. Számos alapértelmezés szerepel még ebben a szkriptben, például a HLAB foglalási állományának helye, vagy az Apache webkiszolgálóval kapcsolatos elérési utak. A szkript „itt-dokumentumok” (here-doc) formájában tartalmazza az Apache biztonságos (SSL) és nem biztonságos kapcsolódáshoz kötődő beállításait, valamint WebLab felhasználók listáját tartalmazó állományt. Ezek az „itt-dokumentumok” lényegében többsoros karaktersorozatok, amelyekben használható a változóbehelyettesítés is. Ezáltal sablonként használhatók. Ezekből a sablonokból előállnak a tényleges tartalmak, amelyek aztán kiíródnak a megfelelő állományokba. Példaként bemutatom a nem biztonságos kapcsolódáshoz kötődő „itt-dokumentumot”:

```
my $apache_vhost_wl_conf_file_content = <<VHWLCFC;
<IfModule mod_fcgid.c>
  SocketPath /tmp/weblab.fcgid_sock
  IdleTimeout 3600
  ProcessLifeTime 7200
  MaxProcessCount 8
  DefaultMaxClassProcessCount 4
  IPCConnectTimeout 8
  IPCCommTimeout 60

  <VirtualHost *:80>
    DocumentRoot ${wl_home_dir}/root
    Alias /static ${wl_home_dir}/root/static
```

```
    <Location /static>
        SetHandler default-handler
    </Location>

    Alias / ${wl_home_dir}/script/weblab_fastcgi.pl/

    <Location />
        Options ExecCGI
        Order allow,deny
        Allow from all
        AddHandler fcgid-script .pl
    </Location>
  </VirtualHost>
</IfModule>
VHWLCFC
```

A teljes többsoros szöveg a `$apache_vhost_wl_conf_file_content` nevű változóba kerül, a változóbehelyettesítések után. Látható, hogy a szöveg határait lényegében a `VHWLCFC` karaktersorozat két előfordulása adja. A szövegben szerepel változóhivatkozás is: `${wl_home_dir}`, amelynek alapértelmezett helyettesítési értéke `/opt/WebLab`, ez pedig éppen az alkalmazás alapértelmezett telepítési könyvtára.

A telepítő viselkedéséből kiemelek néhány lényeges elemet:

- Telepítés előtt törli az esetlegesen már létező telepítési könyvtárat (alapértelmezés szerint ez a `/opt/WebLab`), annak teljes tartalmával együtt. Törli továbbá a `/tmp` könyvtárban esetleg jelenlévő `session` (az alkalmazás session-kezeléséhez szükséges) és `weblab.fcgid_sock` (a FastCGI socket-kezeléséhez szükséges) könyvtárakat tartalmukkal egyetemben. Minderre azért van szükség, hogy biztosan ép telepítést kapjunk.
- Feltételezi, hogy létezik a foglalási állomány (hlab.txt) a HLAB konfigurációt tartalmazó könyvtárban (alapértelmezésben ez a `/etc/hlab`). Ha nem létezik ez az

állomány, akkor kézzel létre kell hozni a megfelelő könyvtárban és (újra) futtatni a telepítőt.

- Telepíti az Apache-hoz nem csak az `fgcid` modult, hanem az SSL kapcsolódáshoz szükséges tanúsítványt is, és az Apache konfigurációját is elvégzi. Telepítés után (újra)indítja a webkiszolgálót, így az alkalmazás elérhetővé válik a http://szerver/ URL-en.

A telepítés menete igen egyszerű:

1. Másoljuk egy könyvtárba a telepítési csomagot jelentő `weblab-`*`verzioszam`*`.tar.gz` állományt (a *verziószám* legyen most 1.0).
2. Tömörítsük ki a csomagot: `tar xzf weblab-1.0.tar.gz`
3. Lépjünk be a `weblab-1.0` könyvtárba: `cd weblab-1.0`
4. Indítsuk el a telepítő szkriptet: `./install.pl`

Ezzel röviden át is tekintettük a telepítési csomag elkészítését, és telepítését. A következő fejezet a felhasználói dokumentációé, amelyben az alkalmazás képességeinek bemutatásáról esik szó olyan mélységben, amely egy átlagos felhasználó számára elegendő ismeretet ad az alkalmazás hatékony használatához.

4. Felhasználói dokumentáció

Jelen dokumentáció tárgyalja az alkalmazás használatbavételének előfeltételeit, az alkalmazás funkcióinak, lehetőségeinek leírását. Ábrák is segítik mindezek szemléltetését. A példák angol nyelvű Internet Explorer 8 (a továbbiakban: IE8) és Mozilla Firefox 3 (a továbbiakban: FF3) böngészőben kerülnek bemutatásra. Ez a dokumentáció feltételezi az alapvető felhasználói ismeretek meglétét.

4.1. Előfeltételek a WebLab alkalmazás használatba vételéhez

A WebLab kényelmes használatához legalább 1024×768-as képernyőfelbontás ajánlott. Ekkor az alkalmazás minden eleme egyszerre látható a képernyőn. A bevitelhez billentyűzet és egér megléte szükséges.

Az alkalmazás használatához JavaScript támogatással rendelkező böngészőre van szükség. Ez manapság nem túlságosan nagy elvárás, minden modern böngésző teljesíti ezt a követelményt. Az alkalmazás a következő böngészőkkel került tesztelésre, amelyekben nagyon hasonló megjelenéssel és funkcionalitással rendelkezett:

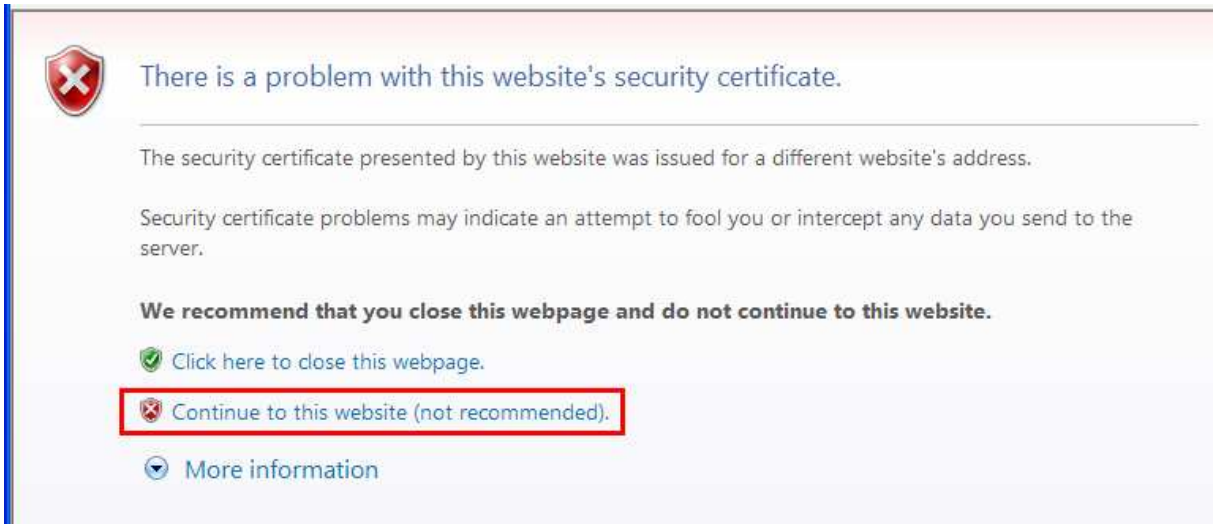
- Firefox 1.5-ös (Linux) és 3-as verzió (Windows)
- Internet Explorer 7-es, 8-as verzió (Windows)
- Opera 10-es verzió (Windows)
- Safari 4-es verzió (Windows)

4.2. Az alkalmazás indítása, bejelentkezés

Indítsuk el a böngészőt és írjuk be a címsorba a HLAB szerver címét.

Ha az IE8 „Certification Error: Navigation Blocked” című hibaüzenetet ad, az abból fakadhat, hogy a HLAB szerveren lévő tanúsítvány érvénytelen adatokat tartalmaz, vagy a tanúsítvány érvényessége lejárt. Ugyanezt a hibajelzést kaphatjuk akkor is, ha hamisított webhelyhez akarunk csatlakozni. Alaposan győződjünk meg arról, hogy nem gépeltük-e el a HLAB szerver címét! Ha mindent rendben találtunk, akkor a „Continue to this website (not recommended).” feliratú hivatkozásra kattintva betöltődik az alkalmazás. Ha pedig nem

bízunk a webhelyben, akkor válasszuk a „Click here to close this webpage” hivatkozást a weboldal bezárásához. Az ábrán pirossal keretezve látható a továbblépést engedélyező hivatkozás.



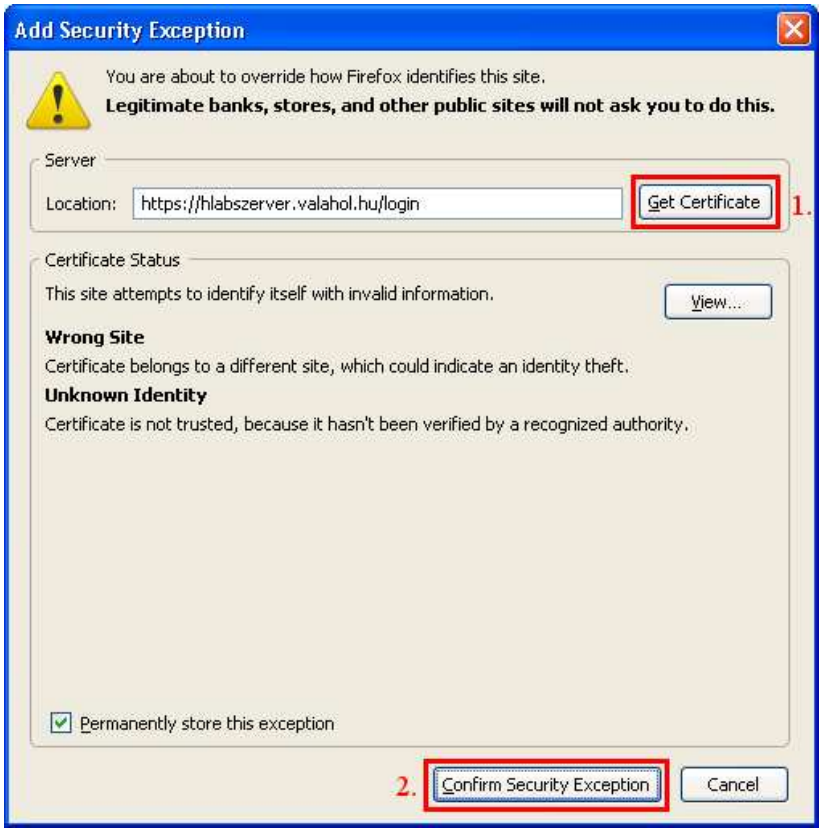
11. ábra

A FF3 pedig „Secure Connection Failed” című hibaüzenetet ad a fenti helyzetben. Az alábbi ábrán piros kerettel kiemelt „Or you can add an exception…” feliratú hivatkozásra kattintva megjelenik két nyomógomb. Ezek közül az „Add Exception” feliratú gomb megnyomásával van lehetőségünk a tanúsítványt elfogadni.

12. ábra

13. ábra

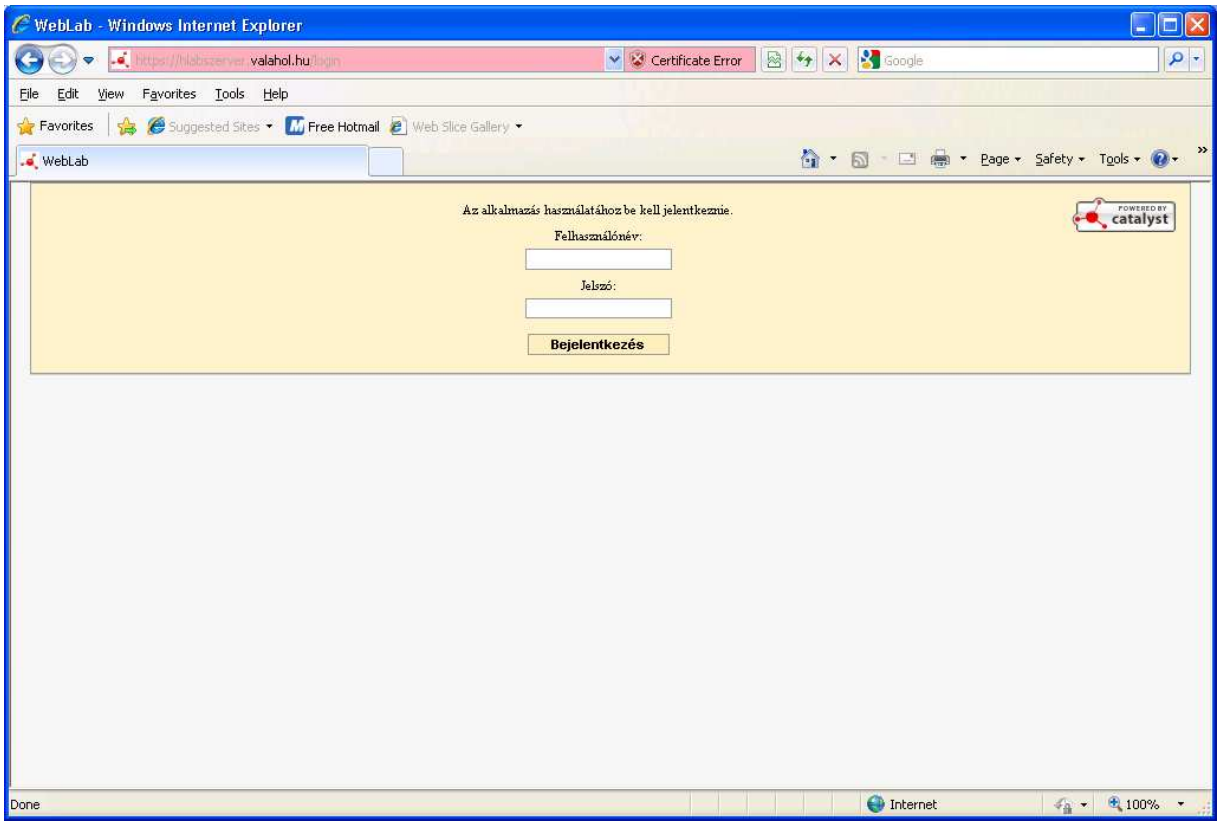
A megjelenő ablakban először kattintsunk a „Get Certificate” majd a „Confirm Security Exception” gombokra.



14. ábra

Ha a „Permanently store this exception” előtt szerepel pipa (alapértelmezés), akkor a FF3 tartósan megjegyzi ezt a tanúsítványt, így következő alkalommal már nem kell újra végigmenni az előző lépéseken.

Miután végigmentünk ezeken a lépéseken, a bejelentkező oldalnak kell megjelennie. A továbbiakban csak az IE8-beli megjelenésről közlünk képernyőképeket, a nagyfokú hasonlóság miatt.

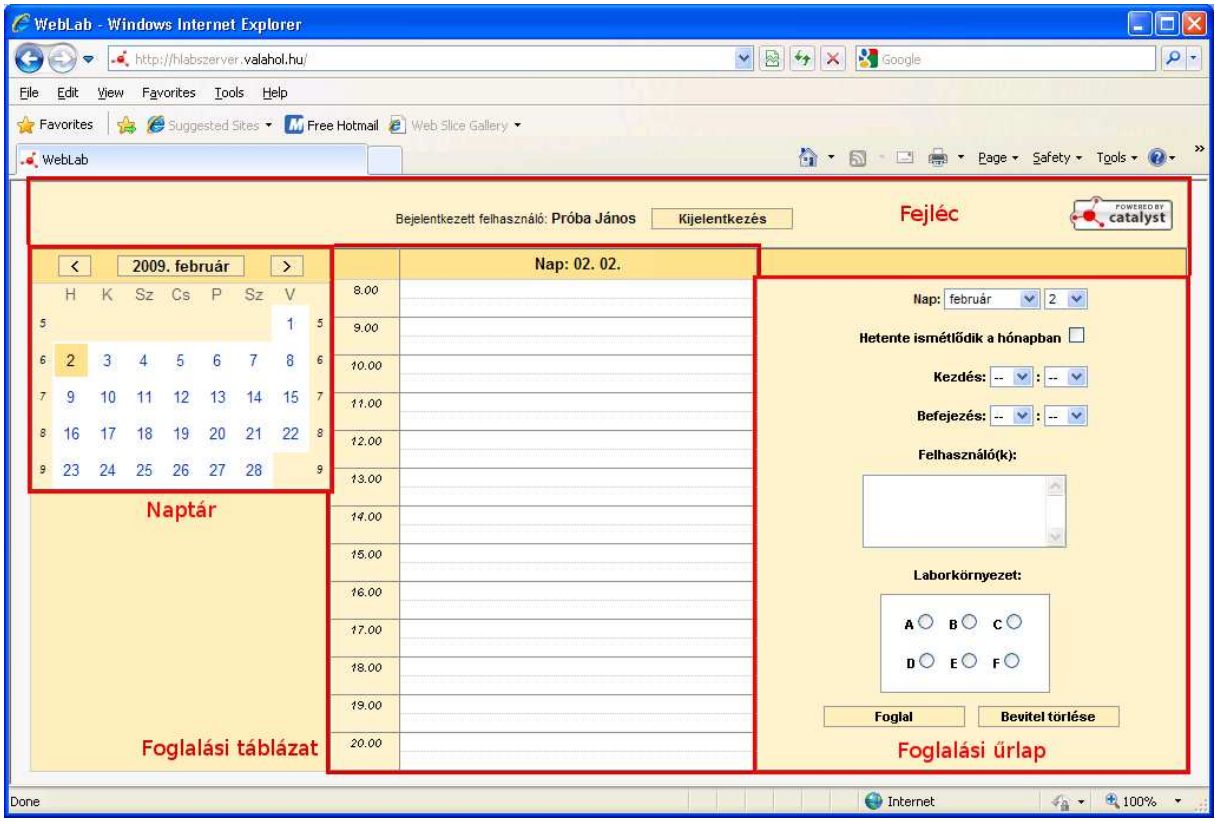


15. ábra

Jelentkezzünk be a HLAB rendszeradminisztrátortól kapott felhasználónévvel és jelszóval. Ha a bejelentkezés többszöri kísérlet után sem sikeres, akkor forduljunk HLAB rendszeradminisztrátorunkhoz a probléma elhárítása érdekében.

Sikeres bejelentkezés után az alábbi ábrán látható oldal jelenik meg (a szemléltetés érdekében kiegészítve az egyes képernyőelemek nevével). A fejlécben láthatjuk nevünket (illetve felhasználónevünket, ha a HLAB rendszeradminisztrátor nem állította be teljes nevünket). Emellett a „Kijelentkezés” gombot találjuk. Ezzel tudunk kilépni az alkalmazásból. Bal

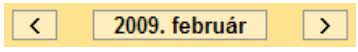
oldalon találjuk a naptárat, középen az adott napra vonatkozó foglalási táblázatot, és jobbra a foglalási űrlapot.



16. ábra

A következőkben az alkalmazás egyes részeit és azok lehetőségeit mutatjuk be részletesebben.

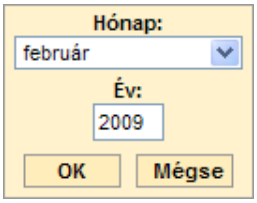
4.3. A naptár

A naptár fejlécében három gombot találunk a navigációhoz: 

A fejléc alatti sorban találjuk a hét napjainak rövidítéseit hétfőtől vasárnapig (H-V).

A bal és jobb szélen a hetek sorszámait láthatjuk kisebb méretű dőlt betűkkel (*1*, *2*, *3*, …).

A bal oldali gombbal az előző hónapra, míg a jobb oldali gombbal a következő hónapra lapozhatjuk a naptárt. A középső gombbal az ábrán látható ablak hozható elő, amelyben legördülő listából kiválaszthatjuk a megfelelő hónapot, illetve begépelhetjük a kívánt évet.



Ily módon a naptár átlapozása könnyebben történhet, ha több hónapot, vagy akár évet kell mozognunk a naptárban.

A naptárban a különböző háttérszínek, illetve a keretezés jelentése:

7	A naptárban kiválasztott nap (narancs háttér, keret nélkül).
8	Ezen a napon legalább egy foglalás történt (kék háttér, keret nélkül).
6	A mai nap (fekete keret).
9	Egyéb nap (fehér háttér, keret nélkül).

4.4. A foglalási táblázat és a foglalási űrlap

A képernyő közepén találjuk a foglalási táblázatot, amely reggel 8 órától este 9 óráig fedi le az adott napot. Egy adott sor 1 óra időtartamnak felel meg, amely segédvonal 30-30 percre oszt. A foglalási táblázat fejlécében a naptárban kiválasztott nap alapján a hónapot és napot láthatjuk, számokkal. Ha például a naptárban 2009. február 14-ét választottuk ki, akkor a **02.14.** felirat jelenik meg a táblázat fejlécében (félkövéren szedve).

Ha a táblázaton belül valamelyik sorra kattintunk, akkor a foglalási űrlapban kitöltésre kerül a kezdés és befejezés időpontja. Alapértelmezés a kétórás foglalás. Ha például a 9 órai sorba kattintunk a segédvonal felett, akkor a kezdés időpontja 9:00, a befejezés időpontja 11:00. Ha ugyanebben a sorba kattintunk a segédvonal alá, akkor a kezdés 9:30-ra, a befejezés 11:30-ra módosul. Természetesen az alapértelmezetten bevitt értékek felülbírálhatók az adatok kézzel történő megadásával.

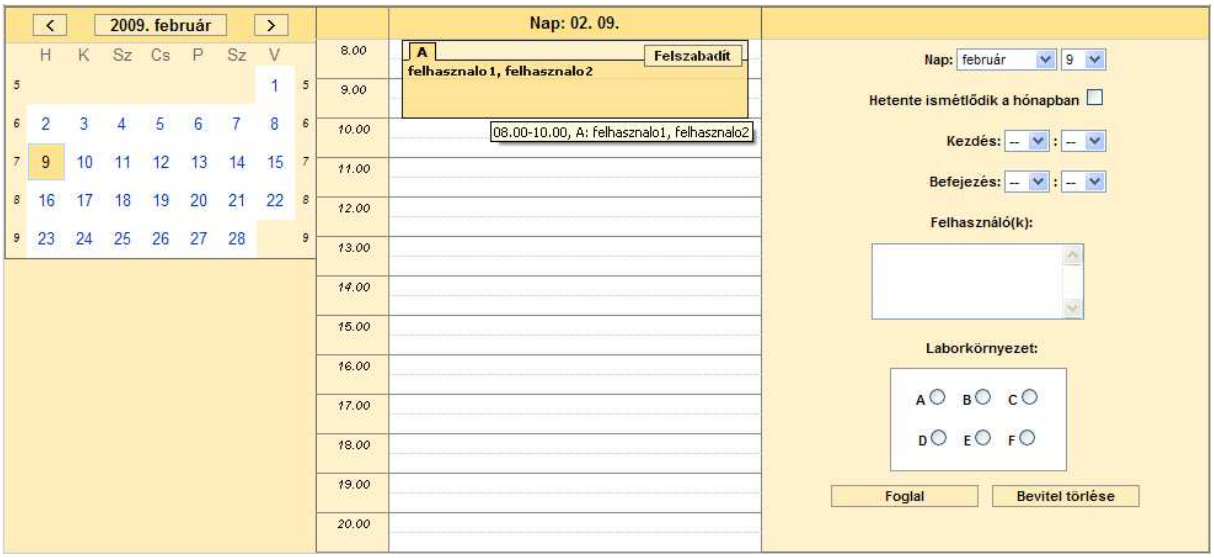
Foglalni a képernyő jobb oldalán található foglalási űrlap segítségével tudunk a még hiányzó adatok bevitelével. Kötelezően megadandóak a következők: Hónap és nap; kezdés és befejezés órája, perce; legalább egy felhasználónév; laborkörnyezet betűjele.

Ha valamilyen kötelező adatot mégis elfelednénk megadni, akkor a „Foglal” gombra kattintás után megjelenő piros keret jelzi a hiányzó adatokat az egyes űrlapelemeknél. Ez megkönnyíti a hiánypótlást. Érvénytelen adatok (a befejezés ideje a kezdésnél korábbi időpont; nem létező nap, például február 30.) esetén szintén piros keret segíti a hibajavítást.

Az űrlapon az adatok megadása a következőképpen történhet: A hónapot, napot, órát és percet legördülő listából választhatjuk ki; a felhasználóneveket a többsoros szövegbeviteli mezőben adhatjuk meg, egymástól legalább egy vesszővel vagy szóközzel elválasztva; míg a laborkörnyezet betűjelét a rádiógombok csoportjából választhatjuk ki.

A „Bevitel törlése” gomb segítségével alapállapotba hozhatjuk az űrlapot. Ekkor a hónap és nap kivételével az űrlapelemekben szereplő összes adat törlésre kerül. Ez a gomb tehát **nem** a foglalás felszabadítására való.

Ha megadtunk minden szükséges adatot, a foglalás megjelenik a foglalási táblázatban, miközben a foglalási űrlap elemeiből törlődnek a bevitt adatok (kivéve a hónapot és napot). A ábra február 9-re történt foglalást mutat 8-10 óra között az A laborkörnyezetre, két felhasználó számára (**felhasznalo1** és **felhasznalo2**).

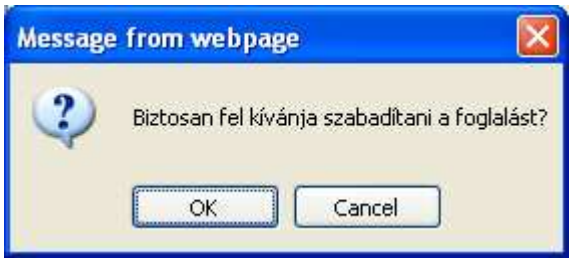


17. ábra

Az ábrán látható gyorsinfó (tooltip) akkor jelenik meg, ha az egérrel elidőzünk egy foglaláson. A gyorsinfó elemei: Kezdés és befejezés ideje, laborkörnyezet betűjele, és a felhasználónevek vesszővel elválasztva.

Ha olyan időszakra veszünk fel foglalást, amelyet a táblázatban már elfoglal egy korábbi foglalás, akkor az adott időszakban szereplő foglalási téglalapban egy új fül jelenik meg, a laborkörnyezet betűjelével címkézve. A füleken kattintva tudunk váltani az egyes lapok között. A lapok egy-egy laborkörnyezethez tartozó foglalás(oka)t tartalmaznak. Az alábbi ábrán látható a lapfüles elrendezés.

Felszabadítani teljes lapot tudunk a „Felszabadít” gombra kattintással. Ekkor a kiválasztott lap (laborkörnyezet) adott időszakra vonatkozó összes foglalását töröljük. A gombra való kattintás után megjelenik az alábbi ábrán látható párbeszédablak, amely rákérdez, hogy valóban fel akarjuk-e szabadítani a foglalást. Az „OK” gombra kattintás a foglalás felszabadítását eredményezi, míg a „Cancel” (Mégse) gombra kattintva nem történik változtatás.



Tipp: Ha rákattintunk egy lapra, az adatai felkerülnek a foglalási űrlapra. Ezáltal nem kell minden adatot kézzel megadnunk, elegendő csak a már meglévő adatokat módosítani, illetve bővíteni.

4.5. Kijelentkezés az alkalmazásból

Ha nem kívánunk tovább dolgozni az alkalmazással, a fejlécben található „Kijelentkezés” gomb segítségével léphetünk ki a WebLab-ból. Biztonsági okokból mindig jelentkezzünk ki munkánk végeztével, illetve ha a gépet őrizetlenül (és lezáratlanul) hagyjuk. Kijelentkezés után ismét a bejelentkező oldal jelenik meg.

Összefoglalás

A következőkben az egyes fő fejezetek szerint haladva készítek összegzést, kitérve a fontosabb megállapításokra, következtetésekre.

Az első fejezetben a fejlesztési környezet kialakításáról számoltam be. Ebben a részben az openSUSE 10.1 Linux alapú HLAB konfiguráció telepítése, beállítása; a fejlesztéshez közvetlen módon kapcsolódó szoftverelemek: SVN verziókövető rendszer, EasyLamp for Eclipse integrált fejlesztői környezet, a Catalyst keretrendszer telepítése, konfigurálása. Mindezeket sikeres végrehajtása tette lehetővé, hogy a fejlesztési munkát hatékonyan elvégezhessem.

A második fejezet a tervezésről szólt. Az alkalmazás általános leírását követően rögzítésre kerültek a kifejlesztendő alkalmazástól elvárt követelmények, a szakterületi fogalmak és folyamatok. A követelmények közül a konzisztencia- és konkurenciakezelés két igen hangsúlyos elem. Mindkettő arra szolgál, hogy az alkalmazás megbízhatóan működjék, abban az esetben is, ha egy időben több felhasználó dolgozik az alkalmazással.

A harmadik fejezetben a tényleges fejlesztés került terítékre. A fejezet az MVC tervezési minta bemutatásával indult. Ezt mindenképpen fontosnak láttam, a megfelelő elméleti-fogalmi megalapozottság érdekében. A következő alfejezetben a választott keretrendszert, a Catalyst-ot mutattam be. Itt röviden említést tettem arról, hogy a Catalyst milyen értelemben használja az MVC tervezési minta Model-View-Controller fogalmait. Ez segítette a keretrendszerben való eligazodást, egyfajta keretet adván.

A fejezet további részében bemutattam, hogyan is áll össze az alkalmazás egy egésszé a sokféle programozási (C, Perl, JavaScript) és leíró (HTML, CSS) nyelv együtteséből. A Template Toolkit sablonnyelv e két kategória határmezsgyéjén található, hiszen dokumentumot írhatunk le segítségével, de használhatunk benne vezérlési szerkezeteket (elágazás, ciklus) is. Kiderült, hogy ezeknek a nyelveknek legalább áttekintő szintű ismerete szükséges a cél eléréséhez. A szerveroldal programozásához használt Perl nyelv, illetve a kliensoldal programozásához használt JavaScript nyelv alaposabb ismerete elengedhetetlen volt.

A fejlesztés utolsó részében a telepítési csomag elkészítéséről esett szó. A kitűzött cél az volt, hogy minél kevesebb fáradtságot jelentsen az alkalmazás telepítése és konfigurálása. A telepítési csomag „formátuma” a Linux-világban elterjedt .tar.gz tömörített állomány.

A tervezésben kitűzött követelmények közül a moduláris felépítés csak részben valósult meg. Ez alatt azt értem, hogy ugyan az alkalmazás valóban több (Perl) modulból épül fel, de ez inkább az MVC tervezési mintának köszönhetően történt így. Viszont mégis teljesíti a bővíthetőség követelményét, ugyanis könnyen felvehető új Controller-ek és bennük Action-ök, amelyek segítségével új funkcionalitással bővíthető az alkalmazás.

A színválasztás előnyben részesítette a halvány háttérszíneket. Összességében kevés szín, és azok árnyalata került használatra. Viszont ahol szükséges volt (például a beviteli hiba jelzésénél), erősebb színt választottam a figyelemfelhívás érdekében.

Az elrendezés is az áttekinthetőséget szolgálja: Balról jobbra haladva egyre több részlettel (több információval) találkozhat a szem.

A Catalyst keretrendszer valóban rugalmasnak bizonyult. Könnyen kiegészíthettem adott feladatosztályokra készített (CPAN) Perl modulokkal.

A FastCGI alkalmazásával az alkalmazás teljesítménye megnövekedett azáltal, hogy nem kell minden egyes kéréskor újabb Perl értelmezőt indítani, valamint betölteni és a memóriában lefordítani a programkódot.

Az elkészült programkód a futtatáshoz szükséges Catalyst keretrendszerrel és egyéb szükséges Perl modulokkal együtt is csupán 13 megabájt körüli tárhelyet igényel a telepítéshez, amely a ma szokásos körülmények között nem jelent nagy erőforrásigényt.

A negyedik fejezetben a felhasználói dokumentáció található. Ez az alkalmazás használójának ad leírást a használati előfeltételekről, az alkalmazás elindításáról, az esetlegesen fellépő tanúsítványproblémákról, a bejelentkezésről, a foglalásról, felszabadításról, és a kijelentkezésről. Ábrák segítik a bemutatottak jobb megértését.

Dolgozatom elérte kitűzött célját, nevezetesen azt, hogy olyan szoftver álljon elő, amely alkalmas arra, hogy a gyakorlatban nap mint nap használják.

Irodalomjegyzék

Könyvek

Laura Lemay – Perl mesteri szinten 21 nap alatt (Kiskapu, 2003.)

Barta Zoltán – Alkalmazásfejlesztés Perlben (Panem, 2005.)

Tom Christiansen, Nathan Torkington – Perl Cookbook, 2nd Edition (O'Reilly, 2003.)

Marcel Gagné – Linux rendszerfelügyelet (Kiskapu, 2002.)

Internetes adatgyűjtés

Magyar nyelvű oldalak

AJAX: http://pentaschool.hu/tavokt/pld_fejezet/Ajax/ajax-alap.php [2010.02.06.]

JSON: http://www.json.org/json-hu.html [2010.02.06.]

MVC (szerző: Janka János):
http://jankajanos.spaces.live.com/Blog/cns!C3E2695FC6F7B0A4!394.entry [2010.02.03.]

YaST: http://hu.opensuse.org/YaST [2010.02.04.]

Angol nyelvű oldalak

Perl programozói dokumentáció: http://perldoc.perl.org/ [2010.02.03.]

EasyEclipse for LAMP: http://www.easyeclipse.org/site/distributions/lamp.html [2010.02.03.]

FastCGI: http://www.fastcgi.com/drupal/ [2010.02.04.]

Catalyst keretrendszer: http://www.catalystframework.org/ [2010.02.03.]

Catalyst kézikönyv: http://search.cpan.org/dist/Catalyst-Manual/lib/Catalyst/Manual/Intro.pod [2010.02.04.]

Subversion: http://subversion.apache.org/ [2010.02.04.]

Template Toolkit: http://template-toolkit.org/ [2010.02.03.]

Open Software License 2.1: http://www.easyeclipse.org/site/licenses/standards/osl-2.1.html [2010.02.03.]

MIT licensz: http://www.opensource.org/licenses/mit-license.php [2010.02.03.]

YUI 2 — Yahoo! User Interface Library: http://developer.yahoo.com/yui/2/ [2010.02.06.]

JavaScript lapfülek (szerző: Patrick Fitzgerald): http://www.barelyfitz.com/projects/tabber/ [2010.02.06.]

Függelék

A függelékben a dolgozatban megjelenő rövidítéseket közlöm, magyarázattal. Természetesen előfordulhat, hogy egy adott rövidítés több dolgot is takarhat, itt azonban csak a dolgozatban képviselt értelmükben szerepelnek.

AJAX (**A**synchronous **J**avaScript **a**nd **X**ML): A webfejlesztésben használatos eljárásgyűjtemény, amely lehetővé teszi, hogy csak az oldal egy meghatározott része kerüljön frissítésre. Az adatcsere a háttérben (aszinkron) módon zajlik.

CD (**C**ompact **D**isc): Optikai tároló, amely hang, kép, mozgókép, illetve adatok tárolására alkalmas. Manapság 700 MB-os kapacitással rendelkezik.

CPAN (**C**omprehensive **P**erl **A**rchive **N**etwork): A Perl programozók kincsestára, rengeteg saját programokban felhasználható Perl modullal.

CSS (**C**ascading **S**tyle **S**heets): Stílusleírás, amely leíró nyelven (például HTML) íródott dokumentum formázására szolgál.

CVS (**C**oncurrent **V**ersions **S**ystem, **C**oncurrent **V**ersioning **S**ystem): Ingyenes verziókövető rendszer. Lehetővé teszi a fejlesztés során korábbi verzióra való visszatérést; fejlesztési ágakra bomlást.

DBI (**D**ata**b**ase **I**ndependent): Perl modul, amely egységes felületet biztosít különböző adatbáziskezelő-rendszerekhez.

DVD (**D**igital **V**ersatile **D**isc, **D**igital **V**ideo **D**isc): Optikai tároló, a CD-vel megegyező méretű korong, de jóval nagyobb tároló kapacitással (például egyoldalas DVD lemez 4,7 GB; egyoldalas, kétrétegű DVD lemez 8,54 GB).

fvwm (**F** **V**irtual **W**indow **M**anager): Ablakkezelő az X Window rendszerhez. Lényegében grafikus környezet Unix/Linux operációs rendszerekhez.

GB (**G**iga**b**yte): Az SI mértékrendszerben 10^9 bájt (azaz 1 000 000 000 bájt). Azonban történeti okokból 1024^3 bájt értelemben is használatos (1 073 741 824 bájt).

GDM (**G**NOME **D**isplay **M**anager): A GNOME grafikus felhasználói felület képernyőkezelője. Könnyen konfigurálható, tetszetős.

HTML (**H**yper **T**ext **M**arkup **L**anguage): A weboldalak leírására szolgáló leíró nyelv.

IDE (**I**ntegrated **D**evelopment **E**nvironment): Integrált fejlesztői környezet. A program szövegének megírásához szükséges szövegszerkesztőn kívül tartalmaz(hat) fordítót, szerkesztőt, nyomkövető, és belövő eszközöket.

IRC (**I**nternet **R**elay **C**hat): kiszolgáló-ügyfél alapú csevegőprotokoll.

ISO lemezkép: Az International Organization for Standardization (ISO) által definiált formátum. Segítségével egy optikai lemez tartalma állományba menthető.

JSON (**J**ava**S**cript **O**bject **N**otation): Szöveg alapú (ember által is olvasható) adatcsere formátum.

LDAP (**L**ightweight **D**irectory **A**ccess **P**rotocol): TCP/IP protokollkészlet felett futó alkalmazásprotokoll, amely címtárszolgáltatás adatainak lekérdezésére és módosítására alkalmas.

MB (**M**ega**b**yte): Az SI mértékrendszerben 10^6 bájt (azaz 1 000 000 bájt). Azonban főleg a memóriáknál 1024^2 bájt értelemben is használatos (1 048 576 bájt).

mc (**M**idnight **C**ommander): Konzolban futó kétpaneles állománykezelő program Unix/Linux rendszerekre.

MIT licensz: A Massachusetts Institute of Technology (MIT) által publikált szabad szoftver licensz. Lehetővé teszi az ilyen licensszel kiadott programok saját programokban való felhasználását, a licensz feltételei szerint (a licenszet a programmal együtt kell terjeszteni).

MS-SQL (**M**icro**s**oft **SQL**): A Microsoft relációs adatbáziskezelő-rendszere.

MVC (**M**odel-**V**iew-**C**ontroller): Tervezési minta, amely az üzleti logikát (Model), a megjelenést (View) és a vezérlést (Controller) különválasztja egymástól.

PID (**P**rocess **id**entifier): A rendszerben lévő folyamatok (processzusok) azonosítására szolgáló (pozitív) egész szám.

RPM (**R**ed Hat **P**ackage **M**anager): Csomagkezelő, amely lehetővé teszi programok telepítését, frissítését, eltávolítását.

SSL (**S**ecure **S**ockets **L**ayer): Biztonságos hálózati protokoll.

SVN (**S**ub**v**ersio**n**): Verziókövető rendszer, hasonló a CVS-hez, de annál fejlettebb.

TT (**T**emplate **T**oolkit): Perl nyelven íródott nyílt forráskódú sablonfeldolgozó rendszer. Segítségével például HTML-oldalakat állíthatunk össze úgy, hogy közben változók értékeit helyettesíthetjük be. Rendelkezik vezérlési szerkezetekkel is.

URL (**U**niform **R**esource **L**ocator): Az Internetes erőforrások szabványosított címe. Megadható az erőforrás eléréséhez használandó protokoll, a célgép címe, a portszám, és az elérési út a célgépen belül.

XML (**E**xtensible **M**arkup **L**anguage): Kiterjeszthető leíró nyelv. A vele leírt dokumentumok lényegében nembináris fa struktúrát követnek.

YaST (**Y**et **a**nother **S**etup **T**ool): Az openSUSE telepítő és beállító eszköze.

YUI (**Y**ahoo! **UI**): Nyílt forráskódú felhasználói felület elemek (például naptár, rendezhető táblázat) JavaScript és CSS alapokon.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani Istennek, hogy megáldotta diplomamunkámat. Hálás vagyok témavezetőm útmutatásaiért, türelméért, amely nélkül e dolgozat nem születhetett volna meg. Továbbá sokat jelentett nekem feleségem bátorítása, amikor elcsüggedtem munkám során. Megköszönök minden támogatást barátoknak és ismerősöknek egyaránt.