****

**EÖTVÖS LÓRÁND TUDOMÁNYEGYETEM**

INFORMATIKAI KAR

[Programozási Nyelvek és Fordítóprogramok Tanszék](http://plcportal.inf.elte.hu/hu)

**Mx DOKUMENTÁCIÓ**

Témavezető: Török Márk

Jakab Árpád Zsolt

Programozó matematikus szak

Esti tagozat

Budapest, 2011. december 06.

Tartalomjegyzék

1. BEVEZETÉS 7

2. Felhasználói dokumentáció 9

2.1. A program rövid leírása 9

2.2. Rendszerkövetelmények 9

2.3. Telepítés 9

2.4. Használat 9

2.5. Bejelentkezés 10

2.5.1. A Home menü 11

2.5.2. A Help menü 11

2.5.3. Administration menü 11

2.5.4. Az Options menü 12

2.5.5. Az Monitoring menü 13

2.5.5.1. Monitoring menü/Manpower charts 13

2.5.5.2. Monitoring menü/Manpower tables 15

2.5.6. Reporting menü 15

2.5.7. A Reporting menü/New Report 16

2.5.7.1. A Reporting menü/Report table 18

2.5.7.2. A Reporting menü/View report 19

2.5.7.3. A Reporting menü/Edit report 19

2.5.7.4. A Reporting menü/Delete report 19

2.5.7.5. A Reporting menü/Restore report 20

2.5.8. A Logout 20

3. Fejlesztői dokumentáció 21

3.1. Követelmények és specifikáció 21

3.2. Használt technológiák 21

3.2.1. A Codeigniter, mint MNV keretrendszer 22

3.2.1.1. A Modellek 23

3.2.1.2. A Nézetek 23

3.2.1.3. A Vezérlők 24

3.3. Az Mx osztályairól általában 24

3.4. Adatbázis 24

3.5. A jelentések felépítése 25

3.6. Jogosultságok 26

3.7. Oldalak 26

3.7.1. Mindenki által látott oldalak 26

3.7.1.1. A Login oldal 26

3.7.1.1.1. A Login vezérlő 27

3.7.1.1.2. A LoginModel modell 28

3.7.1.1.3. A LoginView nézet 28

3.7.1.1.4. A LoginChooseRoleView nézet 28

3.7.1.1.5. A HomeView nézet 28

3.7.1.1.6. A LogoutView nézet 28

3.7.1.2. A Home oldal 29

3.7.1.2.1. A Home vezérlő 29

3.7.1.2.2. A HomeView nézet 29

3.7.1.3. Az Options oldal 29

3.7.1.3.1. Az Options vezérlő 29

3.7.1.3.2. Az OptionsModel modell 30

3.7.1.3.3. Az OptionsView nézet 30

3.7.2. A team member által látott oldalak 30

3.7.2.1. A New report oldal 30

3.7.2.1.1. A NewReport vezérlő 31

3.7.2.1.2. A NewReportModel modell 33

3.7.2.1.3. A NewReportFrameView nézet 33

3.7.2.1.4. A NewReportFirstView nézet 33

3.7.2.1.5. A NewReportSecondView nézet 34

3.7.2.1.6. A ReportTableView nézet 34

3.7.2.2. A View report oldal 34

3.7.2.2.1. A ViewReport vezérlő 34

3.7.2.2.2. A ViewReportModel modell 35

3.7.2.2.3. A ViewReportView nézet 35

3.7.2.3. A Report table oldal 35

3.7.2.3.1. A ReportTable vezérlő 35

3.7.2.3.2. A ReportTableModel modell 36

3.7.2.3.3. A ReportTableView nézet 36

3.7.2.4. A Help oldal 36

3.7.3. Az executive jogú felhasználók által elérhető funkciók 36

3.7.3.1. A New Report oldal 37

3.7.3.2. A View Report oldal 37

3.7.3.3. A Report table oldal 37

3.7.3.4. Az Edit report oldal 37

3.7.3.4.1. Az EditReport vezérlő 37

3.7.3.4.2. Az EditReportModel modell 39

3.7.3.4.3. Az EditReportFrameView nézet 40

3.7.3.4.4. Az EditReportFirstView nézet 40

3.7.3.4.5. Az EditReportSecondView nézet 40

3.7.3.4.6. Az EditReportThirdView nézet 40

3.7.3.4.7. A ReportTableView nézet 41

3.7.3.5. A Delete report oldal 41

3.7.3.5.1. A DeleteReport vezérlő 41

3.7.3.5.2. A DeleteReportModel modell 43

3.7.3.5.3. A DeleteReportFrameView nézet 43

3.7.3.5.4. A DeleteReportFirstView nézet 43

3.7.3.5.5. A DeleteReportSecondView nézet 44

3.7.3.6. A Restore report oldal 44

3.7.3.6.1. A RestoreReport vezérlő 44

3.7.3.6.2. A RestoreReportModel modell 46

3.7.3.6.3. A RestoreReportFrameView nézet 46

3.7.3.6.4. A RestoreReportFirstView nézet 46

3.7.3.6.5. A RestoreReportSecondView nézet 47

3.7.3.7. Adminisztráció 47

3.7.3.7.1. A felhasználók kezelése 47

3.7.3.7.1.1. Az AdminName vezérlő 47

3.7.3.7.1.2. Az AdminNameModel modell 48

3.7.3.7.1.3. Az AdminNameView nézet 49

3.7.3.8. A Monitoring funkció 49

3.7.3.8.1. A Bar Chart oldal 49

3.7.3.8.1.1. A BarChart vezérlő 49

3.7.3.8.1.2. A BarChartModel modell 50

3.7.3.8.1.3. A BarChartView nézet 51

3.7.3.8.2. A Pie Chart oldal 51

3.7.3.8.2.1. A PieChart vezérlő 51

3.7.3.8.2.2. A PieChartModel modell 52

3.7.3.8.2.3. A PieChartView nézet 52

3.7.3.8.3. Monitoring táblázatok 52

3.7.3.8.3.1. A ManpowerTableRelevant oldal 52

3.7.3.8.3.1.1. A ManpowerTableRelevant vezérlő 52

3.7.3.8.3.1.2. A ManpowerTableRelevantModel modell 53

3.7.3.8.3.1.3. A ManpowerTableRelevant nézetei 54

3.7.3.8.3.2. A ManpowerTableAll oldal 54

3.8. Tesztelés 54

3.8.1. Modultesztelés 54

3.8.2. Integrációs tesztek 54

4. ÖSSZEFOGLALÁS 55

5. KÖSZÖNETNYILVÁNÍTÁS 56

6. IRODALOMJEGYZÉK 57

7. CD MELLÉKLET 58

# BEVEZETÉS

Napjaink szoftverfejlesztésének egyik legnagyobb kihívása, hogy pontos előrejelzést tudjunk adni a projektjeink munkaidő, és az ezzel járó költségigényére. Továbbá a fejlesztés vezetőjének szüksége van egy felületre, ahol nyomon tudja követni a fejlesztést, illetve be tud avatkozni annak menetébe, amint az szükséges. Üzleti szempontból viszont ugyanolyan fontos, hogy ne csak a fejlesztés irányítói, de a megrendelők is nyomon tudják követni a projekt állapotait. Ráadásul gyakran előforduló jelenség, hogy a szoftverek megrendelői nem az informatika világában jártas szakemberek, hanem a gazdaság egyéb szegmenseiből érkező üzletemberek. Ezért sem elég önmagában a jó minőségű szoftver elkészítése, ugyanolyan fontos, hogy a megrendelő azon túl, hogy minél pontosabb becslést kap a projekt idő és erőforrás igényéről, folyamatosan nyomon is tudja követni a fejlesztés állapotait, anélkül, hogy a szakmai részletekben el kellene merülnie.

Ahhoz, hogy az előbb felsorolt igényeknek megfeleljen egy eszköz, egyszerre elérhetőnek kell lennie az összes felhasználója számára, olyan szerveren kell futnia, ami hálózaton keresztül elérhető. Munkám során, egy nemzetközi vállalat budapesti fejlesztőközpontjában egy az előbbiekben felsorolt problémákat megoldani képes szoftver, az Mx(*Manpower Planning Reporting Monitoring Controlling Administration System*) implementálásával foglalkozom.

Ez a dokumentum a Mx program prototípusának felhasználói és fejlesztői dokumentációját tartalmazza. A program a fejlesztői csoport számára íródott. A prototípus célja egy olyan rendszer implementálása, mely lehetőséget ad a fejlesztés munkatársainak a munkahelyi tevékenységeik és azok időtartalmának jelentésére; illetve vezetőjének a munkafolyamatok nyomon követésére, irányítására, továbbá segítséget nyújt a számlázási feladatok elvégzésében. Az implementáció alapvető célja egy stabil, a kitűzött elvárásokat maximálisan megvalósító, ugyanakkor felhasználóbarát felülettel rendelkező rendszer készítése volt, amely nyitott a prototípusban még nem, de a fejlesztés későbbi szakaszaiban megvalósítandó funkciók integrálására. A program menüvezérelt, az adatok tárolásához MySQL adatbázist használ.

Az alkalmazás nyújtotta legfontosabb szolgáltatások:

* Munkahelyi riportok készítése, módosítása, törlése a fejlesztés dolgozóinak
* A munkafolyamatok megfigyelése számlázáshoz szükséges összesítések generálása riportokból készített táblázatok és diagramok segítségével
* Adminisztrációs felület a rendszer karbantartására

# Felhasználói dokumentáció

# A program rövid leírása

Az Mx szoftver felhasználói szempontból lényegében egy weboldal, melybe kétféle jogosultságú felhasználóval léphetünk be:

* *team member*
* *executive*

A jogosultságtól függően jeleníti meg a rendszer a felhasználó számára, a felhasználói felületet.

A *team member* jogú felhasználó leginkább jelentéskészítésre tudja használni a programot, ezen kívül még lehetősége van a jelentéseit megtekinteni, illetve egy táblázatban nyomon követni a jelentései havi összesítéseit.

Ezen funkciók az *executive* jogú felhasználó számára is rendelkezésre állnak, de ő ezeken felül még módosíthat, törölhet is jelentéseket, illetve törölt jelentéseket tehet aktívvá. Ezen kívül még elérhető számára az a felhasználói felület, ahol az adatbázis jelentéseit, azaz a munkafolyamatokat grafikonok és táblázatok formájában is nyomon tudja követni. Ez a funkció segíti a számlázást és a munkafolyamatok tervezését is. Ezen felül az adminisztrátori felület szintén az *executive* felhasználó számára érhető el.

# Rendszerkövetelmények

A program futtatásához Windows vagy Linux operációs rendszer, telepített MySQL 5.5 adatbázis-kezelő rendszer Apache 2.2 webkiszolgáló és PHP 5.3 szükséges, melyek megtalálhatóak az alkalmazást is tartalmazó CD-n.

# Telepítés

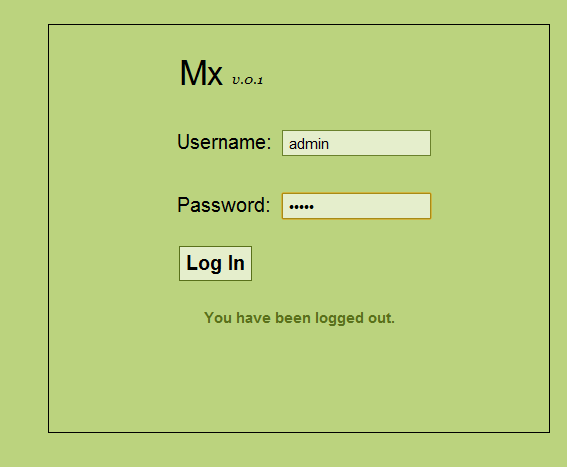
A programot nem kell telepíteni, a merevlemezre telepített webszerver gyökérkönyvtárába másolva használható. Az alkalmazást egy böngésző segítségével érhetjük el.

# Használat

Az alkalmazást két részre érdemes bontani, az egyik, amit általában a *team member* jogosultságú felhasználó használ, illetve egy másik, amit csak az *executive* jogosultságú felhasználó érthet el.

Az alkalmazás teljesen menüvezérelt, a menüpontokat és azok használatát mutatja be a következő alfejezet.

# Bejelentkezés



*A bejelentkezés*

Elsőként egy bejelentkezési ablak fogad minket. A programba belépő felhasználóknak két típusát különböztetjük meg, a *team member*-t, és *executiv*e-ot.

A *executive* joggal a fejlesztés vezetői rendelkeznek, így fel tudnak venni további *team member*-eket, és *executive*-okat.

A bejelentkezéshez tehát felhasználónévre és jelszóra van szükségünk. A program sikertelen bejelentkezés esetén hibaüzenetben tájékoztat minket.

A program felhasználhatóságának szempontjából nagy különbség van az *executive* és a *team member* között. A *team member* nem tud belépni az *Administration és a Monitoring* menüpontokba. Jelentést is csak magának készíthet, és csak a saját jelentéseit és annak táblázatait nézheti meg. Nem módosíthat, és nem törölhet jelentést. Ezzel szemben az *executive* mindenhez hozzáfér, és ezen felül lehetősége van kvázi-*team member*-ként is bejelentkezni, illetve az *Options* menüpontban bármikor válthat a kvázi-*team member* és *executive* nézet között. Erre azért van szükség, mivel a fejlesztés vezetői *team member*-ek is egyben, nekik is ugyanúgy el kell készíteniük a munkahelyi jelentéseiket a számlázáshoz.



*Szellemszerep választás*

# A Home menü

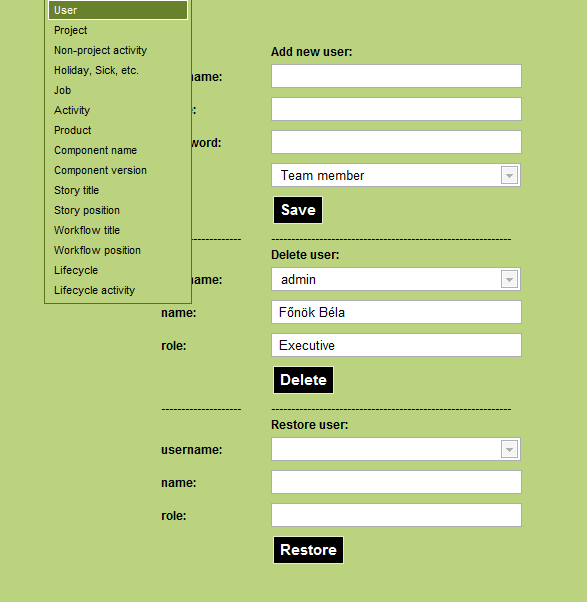
A Belépést követően a felhasználó *Home* menüben találja magát, melyből a *Reporting* menü a jogosultságának megfelelő funkcióit tudja elérni. Ezen kívül a *Home* menüben csak egy üzenet látható, ami közli a felhasználóval, hogy be van jelentkezve, és azt is, hogy milyen néven van bejelentkezve. A *Home* menü a bejelentkezés ideje alatt végig elérhető a *Home* gomb segítségével.

# A Help menü

A *Help* menü szintén közvetlenül indítás után elérhető, melynek használatával rövid információt kaphatunk a program használatáról.

# Administration menü

Jelentkezzünk be *executive*-ként. Csak ekkor érhető el számunkra a *Administration* menüpont, melyben a munkahelyi jelentésünk attribútumainak lehetséges értékeit kezelhetjük. Választhatunk, hogy törlünk (*Delete*), egy törölt értéket visszaállítunk (*Restore*), vagy új értéket veszünk fel (*Save*).

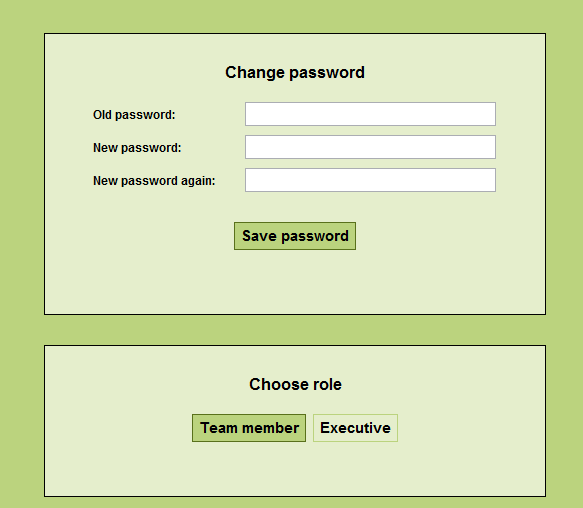


*Adminisztrációs felület*

# Az Options menü

Ha *team member*-ként lépünk be az *Options* menüben a jelszavunkat változtathatjuk meg. Sikertelen változtatás esetén a rendszer hibaüzenetet küld.

Ha *executive* jogosultságú felhasználóval léptünk be, akkor a jelszó megváltoztatásán kívül választhatunk, hogy *executive* vagy *team member* nézetben szeretnénk-e tovább dolgozni. Ha a *team member* nézetet választjuk, akkor pontosan ugyanazokat a funkciókat tudjuk elérni, mint egy *team member* jogosultságú felhasználó, azzal a különbséggel, hogy ezen menüpontban bármikor visszaállhatunk *executive* nézetre, amire természetesen a valódi *team member*-eknek nincs lehetőségük.



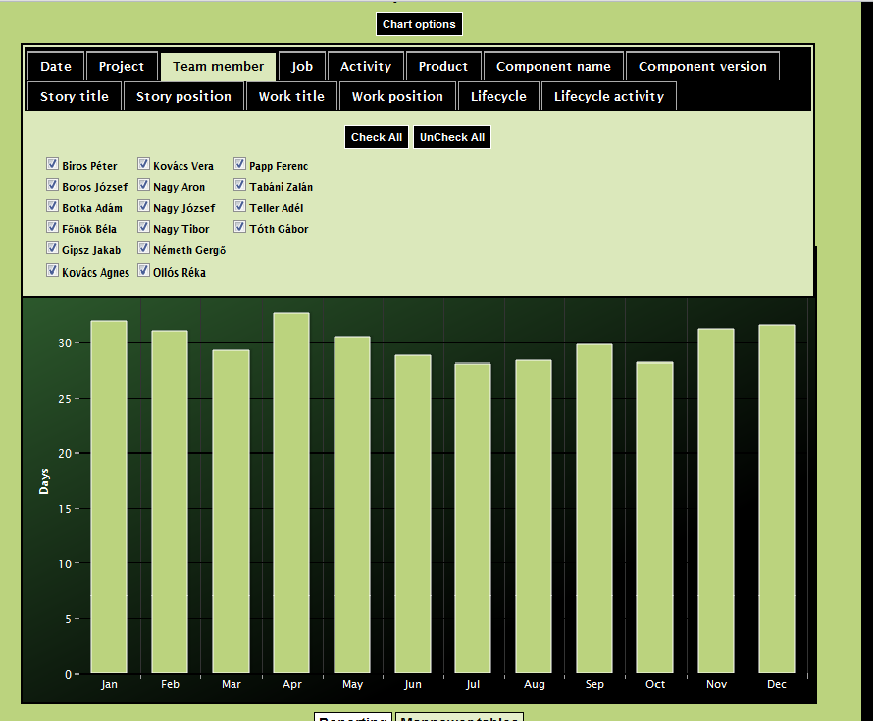
*Beállítások menüpont*

# Az Monitoring menü

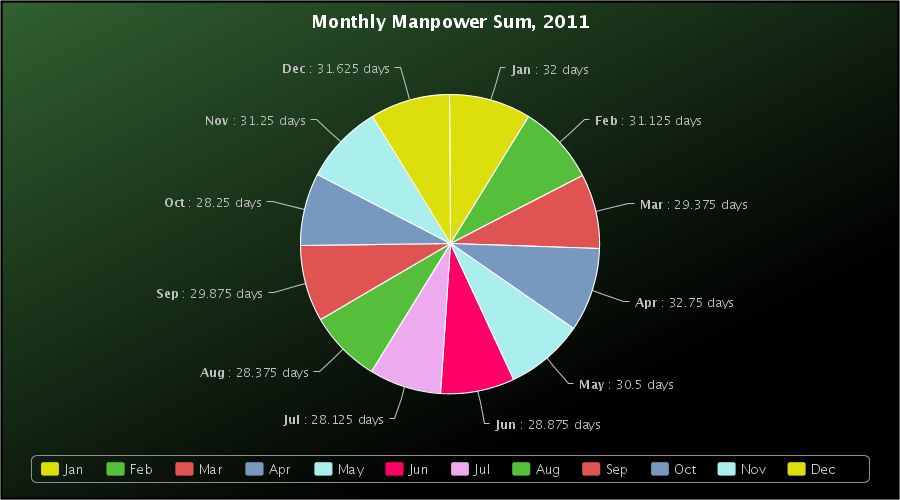
Ebben a menüpontban táblázatokat és diagramokat nézhetünk meg, illetve lehetőségünk van ezeket Excelbe, illetve különböző képformátumokba exportálni. Csak *executive* felhasználók érhetik el ezt a menüpontot. A *team member*-ek nem is látják.

# Monitoring menü/Manpower charts

Ebben diagramokon keresztül követhetjük nyomon a munkatársaink munkafolyamatait, amelyhez az adatokat a munkatársaink jelentései szolgáltatják. A grafikonunkat a jelentések jellemzői alapján finomíthatjuk, exportálhatjuk képformátumokba a végeredményünket.



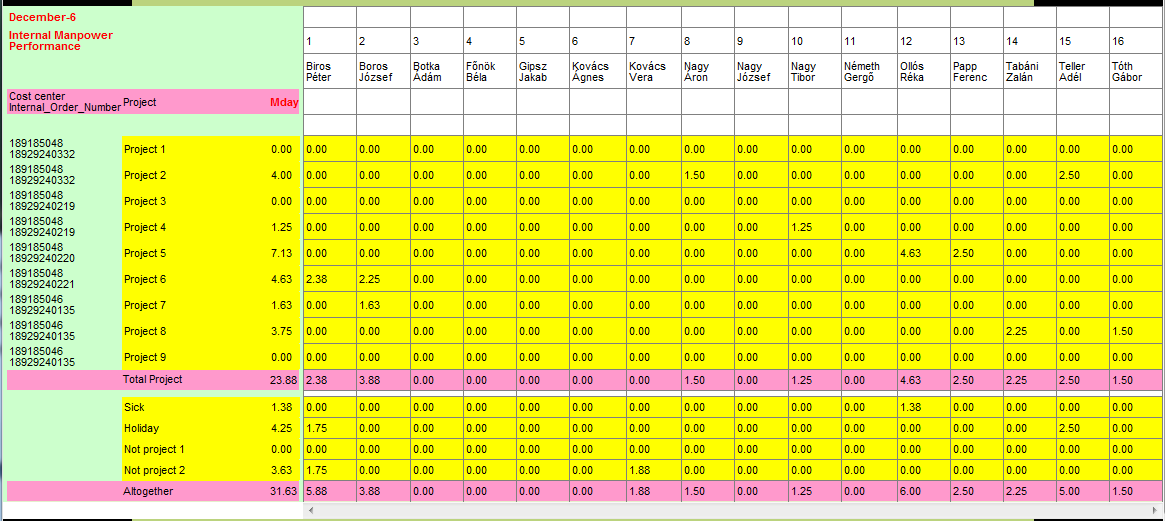
*Grafikus adatmegjelenítés*



*Diagram exportálása képformátumba*

# Monitoring menü/Manpower tables

Ebben táblázatok formájában láthatjuk munkatársaink korábbi hónapokra szóló munkahelyi jelentéseinek összesítéseit, majd exportálhatjuk Excel formátumba a végeredményünket, feltéve, hogy az Excel program telepítve van a gépünkön.

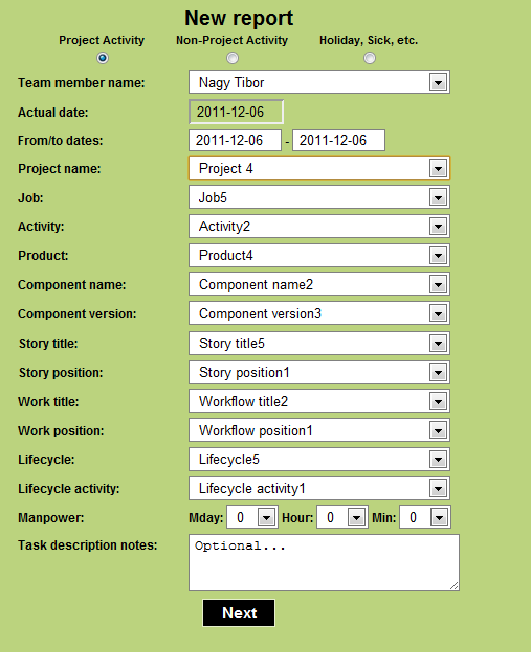


*Munkafolyamat táblázat*

# Reporting menü

A *Reporting* menü lényegében mindennek az alapja. Itt kerül feltöltésre az adatbázis. A *team member* jogosultságúak csak 3 funkciót használhatnak és azt is csak saját magukra. Új jelentést vihetnek fel, megnézhetik a már meglévő jelentéseiket, illetve egy táblázat segítségével nyomon követhetik milyen tevékenységet vittek fel az adott hónapokban. *Executive* joggal is lehet jelentéseket felvinni, megnézni, illetve a tevékenység táblázat is rendelkezésünkre áll, de itt nem csak saját magunkra hajthatjuk végre a funkciókat, hanem a munkatársainkra is. Ezen felül lehetőségünk van még jelentések módosítására és törlésére is egy „*soft delete*” funkció segítségével. Van még egy *Restore report* almenüpont is, ahol a törölt jelentéseinket állíthatjuk vissza.

# A Reporting menü/New Report



*A jelentéskészítő oldal adatkiválasztó nézete*

A *New Report* menüpont egyetlen kivételtől eltekintve, szinte ugyanazt a felületet nyújtja a *team member* és az executive felhasználónak is. A menüpont neve alatt 3 radiogomb található, amelyek segítségével kiválaszthatjuk, hogy milyen jelentést szeretnénk készíteni: „*Project Activity*”, „*Non-Project Activity*” illetve „*Holiday, Sick, etc*„ közül választhat a felhasználó. Ez alatt található a *team member* a saját nevét látja, az *executive* felhasználó viszont egy listát lát, amiből kiválaszthatja, hogy kinek a nevére szeretné a jelentést elkészíteni.

A következő sorban az aktuális dátum látható, alatta pedig megadhatjuk az időszakot, amire a jelentést készítjük felugró kalendáriumok segítségével. Ha nem megfelelő a dátum időszakot próbálunk megadni, a rendszer figyelmeztetést küld. Az ezt követő sorokban legördülő listák segítségével pontosíthatjuk a jelentésünket. Majd megadhatjuk a munkaidőt, amit az általunk korábban kiválasztott tevékenységhez szeretnénk rendelni, ez minimum 15 perc lehet.

A jelentés alján még lehetőségünk van egy jegyzet segítségével rövid leírást írni a jelentésünkhöz. Ha minden kész, akkor a *Next* gombot megnyomva egy ellenőrző felületen találhatjuk magunkat.



*A jelentéskészítő oldal ellenőrző nézete*

Itt már nincs más dolgunk, mint ellenőrizni az adatainkat, és ha minden rendben van, akkor *Save* gombbal menthetünk, de lehetőségünk van editálni is, ekkor az *Edit* gomb visszavisz bennünket a kezdő képernyőre, ahol folytathatjuk a jelentést, vagy a *Cancel* gombbal is visszakerülhetünk a *New report* kezdő képernyőjére, de akkor minden eddigi beállításunk elveszik, helyette az utolsó jelentés adatai jelennek meg, amennyiben volt ilyen. Ha nem, akkor egy alapbeállítás fogad minket. Ha sikeresen elmentettük a jelentésünket, akkor az oldal a *Report table* menüpontba vezérel bennünket.



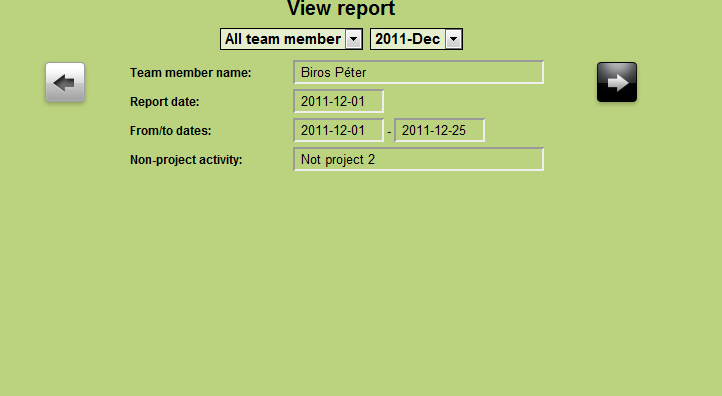
*A jelentések havi összesítő táblázata*

# A Reporting menü/Report table

Ez a menüpont is szinte ugyanazt a felület kínálja a *team member*-nek és az *executive*-nak, azzal a különbséggel, hogy *executive* joggal kiválaszthatjuk, hogy kinek szeretnénk megnézni a táblázatát. Illetve választhatjuk azt is, hogy mindenkiét. A *team member-*nek ez a legördülő menü hiányzik, viszont elérhető egy másik legördülő menü, amiben a hónapot választható ki, aminek a táblázatára kíváncsiak vagyunk. A legördülő menü(k) alatt pedig már maga a táblázat található, aminek sorait a tevékenységek és a hozzájuk tartozó munkaidők találhatóak.

# A Reporting menü/View report

A *View Report* funkció hasonló a *Report table* menüponthoz, az *executive* itt is két legördülő listával navigálhat a jelentések között, egy a neveket és egy a hónapokat tartalmazóval. Ezek lényegében ugyanazok, mint a *Report table* listái. Alatta már a korábban bevitt jelentések találhatóak, amelyek között két az oldal szélein található nyilak segítségével navigálhatunk.



*Jelentések megtekintésére szolgáló oldal*

# A Reporting menü/Edit report

Ez a funkció csak *executive* joggal érhető el. Itt van lehetőségünk a jelentéseink megváltoztatására. A kezdő oldal lényegében ugyanaz, mint a *View report* funkciónál, csak itt található egy Edit gomb, amivel - miután kiválasztottuk a kívánt jelentést - tovább léphetünk az editáló oldalra. Innentől meg lényegében minden úgy működik, mint a *New report* funkciónál. *Save* után itt is a *Report table* oldalra kerül a felhasználó.

# A Reporting menü/Delete report

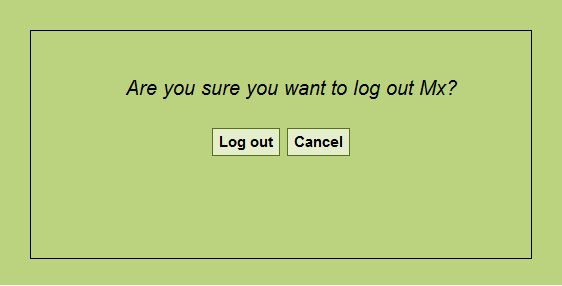
Ez a funkció is csak *executive* felhasználóknak áll rendelkezésére. Ennek segítségével tudjuk törölni a jelentéseket az aktív jelentések közül. A kiválasztás ugyanúgy működik, mint az *Edit report* esetében, csak itt *Save* helyett *Delete* gombot kell nyomnunk, ami törli az aktív jelentések közül az általunk kijelöltet.

# A Reporting menü/Restore report

Ez a funkció is csak *executive* felhasználóknak áll rendelkezésére. Ennek segítségével tudjuk törölt jelentéseket újra aktívvá tenni. A funkció lényegében a *Delete* fordítottja, minden ugyanúgy működik, csak *Delete* helyett *Restore* gombot kell nyomni.

# A Logout

A felső menünk jobb szélén található egy *Logout* gomb, amit megnyomva megjelenik egy oldal, amelyen ismét *Logout* gombot választva kiléphetünk, *Cancel* gombot nyomva pedig visszatérünk az Mx rendszerbe pontosan arra az oldalra, ahonnan indítottuk a kijelentkezést, ugyanazokkal a beállításokkal.



*Kijelentkezés*

# Fejlesztői dokumentáció

# Követelmények és specifikáció

A cél egy munkaidő jelentések készítésére, a jelentések kezelésére és exportálására és irányítására használható rendszer, ami tovább bővíthető egy olyan rendszerré, amely lehetővé teszi a munkafolyamatok tervezését is.

A következő követelményeket fogalmaztuk meg:

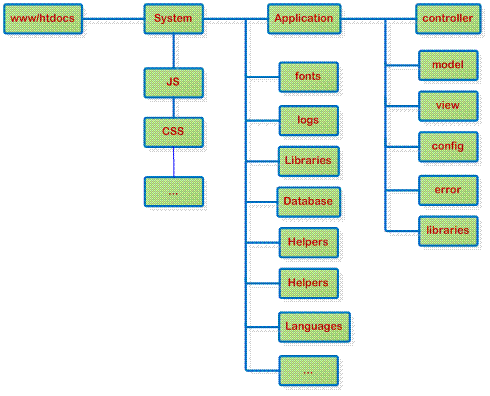
* belső intraneten legyen elérhető;
* képes legyen akár 100 felhasználót is egyszerre kiszolgálni a rendszer;
* egy hétköznapi böngészővel legyen elérhető, lehetőleg egyéb komponensek telepítése nélkül;
* az adatokat adatbázisban tárolja, és az adatbázis PostgreSQL vagy MySQL legyen.

A program készítése során a következő szempontokat vettük figyelembe:

* könnyű kezelhetőség;
* továbbfejleszthetőség, legyen nyílt a rendszer;
* grafikus felhasználói felület;
* újrafelhasználható komponensek;
* agilis szoftverfejlesztés.

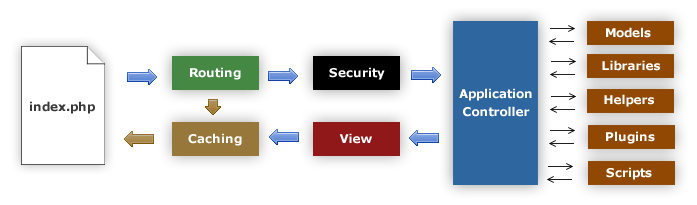
# Használt technológiák

Ezek a célok mind megvalósíthatóak PHP nyelven a Codeigniter keretrendszer segítségével. Továbbá a Codeigniter sok korszerű fejlesztési eszközt támogat.



*A Codeigniter keretrendszer könyvtárszerkezete*

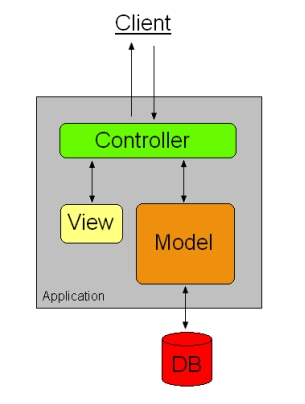
# A Codeigniter, mint MNV keretrendszer



*A keretrendszer működése*

Az általam használt keretrendszer, a Codeigniter az MNV szerkezeti mintát követi. A felhasználó által látogatott minden oldal mögött egy *Model-View-Controller* hármas áll.A fejlesztőnek nincs lehetősége ettől eltérni, a Codeigniter szerkezete is olyan, hogy külön mappákban tartalmazza a modelleket, a nézeteket és a vezérlőket.

Ily módon majd minden oldalhoz tartoznia kell minimum egy vezérlőnek, egy modellnek és egy nézetnek. Előfordulhatnak olyan oldalak is, melyek nem tartalmaznak adatbázis lekérdezéseket, ezért modellre sincs szükségük, de nézetet és vezérlőt ők is tartalmaznak.



*Az MNV szerkezeti minta működése*

# A Modellek

Minden modell a Codeigniter-ben a *CI\_Model* ősosztályból származik egyszerű öröklődéssel. Az Mx-hez írt modellek elsősorban a Codeigniter *Active record* osztályának függvényeit hívják meg, melyek lekérdezéseket adnak át beállított adatbázis számára. Ez jelen esetben MySQL, de bármi más is lehetne, így bármikor kicserélhető az Mx alatti adatbázis MySQL-ről PostgreSQL-re.

# A Nézetek

Az *Mx* nézetei PHP, HTML és JavaScript nyelveken íródtak. Továbbá JavaScript oldalról JQuery, JQueryUI és Highcharts és könyvtárakat is integráltam. A JQuery egy népszerű [JavaScript keretrendszer](http://hu.wikipedia.org/wiki/JavaScript_keretrendszer), amely támogatja az Ajax webfejlesztési technikát, amivel a lapújratöltés nélküli kliens szerver kommunikációt valósítottam meg.

A JQueryUI pedig egy JQuery felhasználói interfész könyvtár, amely elsősorban JavaScript és CSS fájlokra épül. A Highcharts pedig egy olyan JavaScript könyvtár, amelyet interaktív grafikonok készítéséhez használhatunk. A nézetek mindig közvetlenül a vezérlőkkel kommunikálnak.

# A Vezérlők

A vezérlők a *CI\_Controller* ősosztályból egyszerű öröklődéssel származtatott osztályok. Az eseményeket jellemzően felhasználói interakciókat dolgozzák fel és válaszolnak rájuk. Rajtuk keresztül kommunikálnak a modellek és a nézetek is. Még egy nézet Ajax hívásai is a vezérlőn keresztül jutnak el az adott modellig. Minden vezérlőnek van egy *index()* eljárása. Ez hívódik meg közvetlenül a konstruktor után. A legtöbb eljárást és függvényt általában az *index()* hívja meg. A vezérlők a szükséges változóikat a *data* tömb indexeiben tárolják, szükség esetén ezt a tömböt adják tovább a nézeteknek. A tömb a nézetekben önmagában már nem, csak indexeiben jelenik meg. Így logikailag a tömb indexei is változónak tekinthetők, ezért az egyszerűség kedvéért a változóként fogok hivatkozni a tömb indexeire.

# Az Mx osztályairól általában

A program osztályait a modellek és a vezérlők adják, melyek a *CI\_Model*, illetve a *CI\_Controller* osztályokból származnak. Az osztályok konstruktoraiban közös, hogy mind meghívja az ősosztálya konstruktorát, illetve a vezérlők még betöltik a hozzájuk tartozó modellt is, ezért amennyiben egy konstruktor mást nem csinál, a jövőben eltekintek a működésének külön részletezésétől. Minden ilyen osztálypároshoz hozzárendelhető egy funkció, amit ellát. Az egyes osztályokat a hozzájuk tartozó oldalak fejezeteiben fejtem ki.

# Adatbázis

1. A program jelen verziója 14 adattáblát használ:

* activity(validity, activity\_id, activity)
* comp\_name(validity,comp\_name\_id, comp\_name)
* comp\_version(validity,comp\_version\_id, comp\_version)
* job(validity, job\_id, job)
* lifecycle(validity, lifecycle\_id, lifecycle)
* lifecycle\_act(validity, lifecycle\_act\_id, lifecycle\_act)
* name(validity, name\_id, username, userpass, name, role)
* product(validity, product\_id, product)
* project(validity, project\_id, project, cost\_center, internal\_order)
* story\_pos(validity, story\_pos\_id, story\_pos)
* story\_title(validity, story\_title\_id, story\_title)
* work\_pos(validity, work\_pos\_id, work\_pos)
* work\_title(validity, work\_pos\_id, work\_pos)
* job\_report(validity, report\_id, jr\_name\_id, date, from\_date, to\_date, task\_desk, jr\_project\_id, jr\_job\_id, jr\_activity\_id, jr\_lifecycle\_id, jr\_lifecycle\_act\_id, jr\_product\_id, jr\_comp\_name\_id, jr\_comp\_version\_id, jr\_story\_title\_id, jr\_story\_pos\_id, jr\_work\_title\_id, jr\_work\_pos\_id, manpower)
  1. Az adatbázis tábláiból sosem törlünk adatot, pusztán az adott sor *validity* mezőjét állítjuk hamisra.

# A jelentések felépítése

Az Mx alkalmazás lényegében a munkaidő jelentésekre épül. Háromféle munkaidő jelentést különböztetünk meg:

* projekt aktivitás: ilyenkor a jelentés munkaideje egy konkrét projektre írható;
* nem projekt aktivitás: ilyenkor a jelentés munkaideje nem számolható el projektként;
* egyéb: szabadság, betegség, stb.

A nem projekt aktivitás és az egyéb munkaidő jelentés jellemzői:

* *Team member name;*
* *Actual date;*
* *From/to dates;*
* *Non-project activity;*
* *Manpower;*
* *Task descreption notes.*

A projekt aktivitás munkaidő jelentés jellemzői:

* *Team member name;*
* *Actual date;*
* *From/to dates;*
* *Project name;*
* *Job;*
* *Activity;*
* *Product;*
* *Component name;*
* *Component version;*
* *Story title;*
* *Story position;*
* *Work title;*
* *Work position;*
* *Lifecycle;*
* *Lifecycle activity;*
* *Manpower;*
* *Task descreption notes.*
  1. A jelentéseket a *job\_report* táblában tároljuk.

Az adatbázis szintjén minden jelentésnek ugyanolyan attribútumai vannak, mint a projekt aktivitás jelentésnek. Csak a felhasználói felületen jelenítjük meg másként.

# Jogosultságok

1. Az egyes felhasználók a *name* táblában vannak elmentve. A *name* tábla *role* mezője mutatja az adott felhasználó jogosultságát.

Két fajta jogosultságot különböztetünk meg:

* *team member*, értéke az adatbázisban 1;
* executive, értéke az adatbázisban 0.

Arról, hogy a felhasználó a számára fenntartott oldalakhoz férhessen hozzá alapvetően a *MenuView* nézet gondoskodik, ami majd minden oldalba integrálva van. Ezen kívül minden nézetben van egy ellenőrzés, ami figyeli a felhasználó jogait.

# Oldalak

Az oldalakat logikailag három részre lehet osztani. A mindenki által látott, a *team member* által látott és az *executiv*e által látott oldalakra.

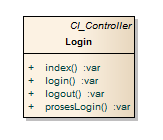
# Mindenki által látott oldalak

Ezek azok az oldalak, amelyek mindenki számára elérhetőek. Ebben az esetben a *team member* és az *executive* fizikailag is ugyanazt az oldalt éri el.

# A Login oldal

Ez az oldal végzi a be- és kiléptetést, továbbá a jogosultságok ellenőrzését.

# A Login vezérlő

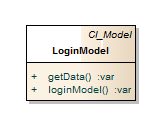
****

*A Login osztálydiagramja*

A *Login* funkció vezérlő osztálya, eljárásai:

* **public function** login(): A konstruktor betölti az *url és a form* helpereket*; a from\_validation, a session* könyvtárakat*; és a LoginModel* modellt;
* **public function** *index():* betölti a *LoginView* nézetet;
* **public function** proseslogin(): végrehajtja a beléptetést. Létrehoz egy munkamenetet, amelyben a modellből kapott felhasználóval kapcsolatos értékeket (név, felhasználónév, felhasználó azonosító, szerep) tárolja egészen a kilépés végrehajtásáig. Ezen kívül eltárol még egy *ghostrole* azaz szellemszerep nevű változóban egy értéket is, ami kezdetben megegyezik a felhasználó szerepével. Ha a modellből nem kapott adatot, üzenetet küld a *LoginView* nézetnek: „*!! Wrong Username or Password !!”.* Ellenkező esetben ellenőrzi a felhasználó szerepét. Ha az *executive*, akkor betölti a *LoginChooseRoleView* nézetet és a szellemszerepet a felhasználó által választottra állítja, majd betölti a *HomeView* nézetet. A felhasználó a szerepe és a szellemszerepe alapján fér hozzá a funkciókhoz. *Az executive* jogosultságú funkciókhoz csak akkor fér hozzá, ha a szerepe és a szellem szerepe is *executive* jogosultságú, minden más esetben *team member* jogosultságú funkciókhoz kap hozzáférést. A *team member* szellemszerepe természetesen konstans *team member*;
* **public function** logout(): Betölti a *LogoutView* nézetet, ha a felhasználó a *Logout* gombra kattint, megszünteti a munkamenetet és kiléptet, majd betölti a *LoginView* nézetet.

# A LoginModel modell



*A LoginModel osztálydiagramja*

A *Login* funkció modell osztálya. Egy függvénye van:

* **public function** getData() : a függvény a *name* tábla azon sorait kérdezi le, amelyekre a *username* és a *userpass* attribútumok értéke megegyezik a felhasználó által írt felhasználónév és jelszó párossal, és a *validity* attribútum igaz, azaz a felhasználó nincs törölt státuszban;

# A LoginView nézet

A *Login* funkció bejelentkezési nézete. Egy-egy bemeneti mezőt tartalmaz a felhasználónévnek és a jelszónak. Kilépésnél és sikertelen belépésnél megjeleníti a vezérlő üzeneteit.

# A LoginChooseRoleView nézet

Két gomb található ezen a nézeten, *Team member* és *Executive*. Ezekkel választhat az *executive* szerepű felhasználó, hogy milyen szellemszereppel kíván belépni.

# A HomeView nézet

A kezdőoldal nézete. Közli a felhasználóval, hogy milyen néven lépett be. A nézet tetején található egy felső menü, amiből a *Reporting* menüfunkcióit tudjuk elérni a jogosultságunknak megfelelően.

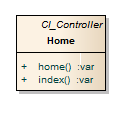
# A LogoutView nézet

A kilépés nézete. Két gomb közül választhatunk rajta. A *Log out* gombhatására a *Login* vezérlő *logout()* függvénye megszünteti a munkamenetet és kiléptet, majd betölti a *LoginView* nézetet. A *Cancel* gomb hatására meghívódik a *window.history.go()* JavaScript függvény -1 paraméter értékkel.

# A Home oldal

A *Home* az Mx kezdőoldala. Ehhez az oldalhoz nem tartozik modell, csak nézet és vezérlő.

# A Home vezérlő



*A Home osztálydiagramja*

*Home* funkció vezérlő osztálya. Egy eljárása van:

* **public function** Index(): betölti a *HomeView* nézetet és elküldi neki üzenetben a felhasználó nevét;

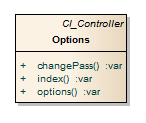
# A HomeView nézet

A kezdőoldal nézete. Közli a felhasználóval, hogy milyen néven lépett be. A nézet tetején található egy felső menü, amiből a *Reporting* menüfunkcióit tudjuk elérni a jogosultságunknak megfelelően.

# Az Options oldal

Ez az oldal felel a jelszóváltoztatásért és az *executive* szellemszerep váltásáért.

# Az Options vezérlő

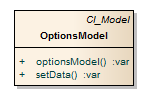


*Az Options osztálydiagramja*

Az *Options* menü vezérlő osztálya. Eljárásai:

* **public function** index(): betölti az *OptionsView* nézetet. A függvény az *OptionsView* *Save* gombjának megnyomásakor meghívja a *changePass()* függvényt. Az *OptionsView* *Team member* gombjának megnyomásakor a szellemszerepet *team member* értékre állítja. Az OptionsView *Executive* gombjának megnyomásakor pedig a szellemszerepet *executive* értékre állítja. Ha nem megfelelő működést tapasztal, üzenetet küld az *OptionsView* nézetnek;
* **public function** changePass(): ellenőrzi a régi jelszó és az új jelszavak bemeneti mezőit, és ha azok megfelelnek a feltételnek, akkor meghívja az *OptionsModel* *setData()* függvényét, ezzel megváltoztatva a jelszót.

# Az OptionsModel modell



*Az OptionsModel osztálydiagramja*

Az *Options* oldalmodell osztálya. Egy függvénye van:

* **public function** setData(): A *name* táblában felülírja a jelszó attribútumának értékét.

# Az OptionsView nézet

Az *Options* nézete. Három bemeneti mezőből áll, egy a régi jelszó és kettő az új jelszó számára. Ezen kívül, van benne egy ellenőrzés, ami a felhasználó szerepét figyeli, és ha az *executive*, akkor felépít még két HTML gombot. Ezek mutatják, hogy éppen milyen szellemszereppel rendelkezik az egyébként *executive* felhasználó, illetve ezek segítségével válthat szellemszerepet. A nézet alján linkek találhatóak különböző oldalakhoz, szellemszereptől függően.

*Executiv*e számára: *Reporting, Manpower tables, Charts.*

*Team member* számára: *New report, View report, Report table*.

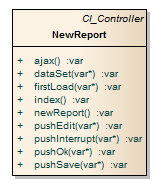
# A team member által látott oldalak

Ezek az oldalak csak *team member* szellemszereppel érhetőek el.

# A New report oldal

Ez az oldal felel az új jelentések beviteléért.

# A NewReport vezérlő



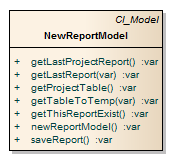
*A NewReport osztálydiagramja*

A *New report* funkció vezérlő osztálya.

A következő eljárásai vannak:

* **public function** newReport(): a konstruktor betölti a *NewReportModel* és a *LoginModel* modelleket;
* **public function** index(): meghívja a *dataSet()* eljárást, ami a *data* tömb kezdeti értékét állítja be. Majd meghívja a *firstLoad()* eljárást, ami a lényegében az oldal inicializálásáért felelős. A funkció nézeteinek különböző gombjainak hatására meghívja a szükséges segédeljárásait. Betölti a *NewReportFrameView* nézetet, ami pedig a *change\_view* változóban tárolt értéknek megfelelő nézetet tölti be. A változó értéke kezdetben 0;
* **public function** dataSet(&$data): a *data* tömb kezdeti értékeit állítja be. Ehhez meghívja a *NewReportModel* szükséges függvényeit és beolvassa a *NewReportFirstView* nézet legördülő listájának értékeit. Az első betöltést tároló változót hamisra állítja. Ez a változó akkor válik igazzá, amint lefut a *firstLoad()* függvény, ez pedig csak az első oldalbetöltésnél fut le. A további oldalbetöltéseknél a változó értéke hamis marad. Ennek a változónak a segítségével állítja be a nézet a legördülő listáinak kezdeti értékeit. A jelentés típusát kezdeti értékre állítja, ami "*pa"*, azaz projekt aktivitás;
* **public function** pushOk(&$data): a *NewReportFirstView* nézet *Next* gombjának megnyomására hívja meg az *index()* eljárás. Az eljárás ellenőrzi, hogy a jelentés létezik-e már, a munkaidő nem 0-e, illetve a jelentés kezdeti és végdátuma egy hónapba esik-e. Ha minden megfelelő, akkor jelentés típusát beállítja a felhasználó által kijelölt rádiógombnak megfelelően. A *change\_view* változó értékét 1-re állítja, aminek hatására betöltődik a *NewReportSecondView* nézet;
* **public function** pushSave(&$data): a *NewReportSecondView* nézet *Save* gombjának megnyomásakor hívja meg az *index()* eljárás. Ellenőrzi, hogy nem az *Ok* gomb megnyomásával és böngészőkbe ágyazott vissza gomb segítségével jutott ide a felhasználó, így kizárja a lehetőségét annak, hogy újra elmenthessük már meglévő jelentésünket. Ha a felhasználó mégis ezzel próbálkozik, a *change\_view* értékét 0-ra állítja, visszatölti a kezdeti nézetet és hibaüzenetben figyelmezteti a felhasználót: „*The Job report exist!*”. Ha minden rendeltetésszerűen zajlik, meghívja a *NewReportModel saveReport()* függvényét ezzel elmentve az adatbázisba a jelentést. Ezután betölti a *ReportTableModel* modellt. Meghívja a *getReports()* és a *getProjects()* függvényeit és elmenti ezeket egy-egy változóba. A *change\_view* változót 2-re állítja ezzel betölti *ReportTable* nézetet, ami a jelentéseknek egy táblázata;
* **public function** pushEdit(&$data): a *NewReportSecondView* nézet *Edit* gombjának megnyomásakor hívja meg az *index()* eljárás. Elmenti a jelentés típusát, és a *change\_view* változót 0-ra állítja. Újra az első nézetet tölti be a felhasználónak, így módosíthatja a jelentését;
* **public function** pushInterrupt(&$data): A *NewReportSecondView* nézet *Cancel* gombjának megnyomásakor hívja meg az *index()* eljárás. Megszakítja a jelentés bevitelét, betölti a *NewReportFirstView* nézetet annak inicializáláskori értékeivel, azaz az utolsó mentet jelentés értékeire állítja a legördülő listáit;
* **public function** firstLoad(&$data): az első oldalbetöltésnél hívja meg az *index()* eljárás. A *first\_load* változót igazra állítja. Meghívja a *NewReportModel* modell *getLastProjectReport()* függvényét, ami az utolsó projekt jelentés adatait tartalmazza. Ha volt utolsó jelentés az adatbázisban, akkor ezt adja tovább a *NewReportFirstView* nézetnek. Ha nem, akkor -1-ekkel tölti fel az tovább adott azonosítókat, így mivel minden azonosító nem negatív egész szám, ezért nem fognak megjelenni a legördülő menükben;
* **public function** ajax(): betölti a *NewReportModel* modellt, majd a *NewReportFirstView* Ajax hívását továbbítja neki. Ezután elvégez egy vizsgálatot a nézettől átvett paraméter értékén. A paraméter értékében a projekt típusa van tárolva, ami alapján meghívja a modell megfelelő függvényét, ami a hívás paraméterének értéke alapján adja vissza az utolsó mentett jelentést, ami tehát vagy az utolsó mentett projekt aktivitás, vagy nem projekt aktivitás vagy egyéb jelentés lehet. Ezután egy elágazás megvizsgálja, hogy nem üres-e a modelltől visszakapott tömb. Ha igen, akkor ugyanúgy, ahogy a *firstLoad()* eljárás, ez is -1-ekkel tölti fel a nézetnek tovább adott azonosítókat. Végül visszaadja a külső hívásnak a modelltől kapott tömböt.

# A NewReportModel modell



*A NewReportModel osztálydiagramja*

A *New report* funkciómodell osztálya. Függvényei:

* **public function** getProjectTable(): visszaadja a project tábla sorait;
* **public function** getTableToTemp($value): visszaadja a paraméterének megfelelő tábla sorait;
* **public function** getLastProjectReport(): az utolsó projekt aktivitást adja vissza;
* **public function** getLastReport($data): az utolsó paraméterének megfelelő jelentést adja vissza;
* **public function** getThisReportExist(): visszaadja, hogy a felhasználó által bevitt jelentés létezik-e;
* **public function** saveReport(): menti a felhasználó által bevitt jelentést.

# A NewReportFrameView nézet

A központi nézet, ez tölti be a többi nézetet a *change\_view* változó értékének megfelelően.

# A NewReportFirstView nézet

A *New report* editálásért felelős nézete.

Ez a nézet fogadja a felhasználót, ahol a jelentésének típusát és jellemzőit állíthatja be rádiógombok és legördülő listák segítségével, amelyek kezdeti értéke az utolsó azonos jelentés, ha létezik olyan. A rádiógombok változását, melyekkel a jelentés típusát állítjuk be JavaScript figyeli, és Ajax hívások segítségével kéri le a szükséges adatokat. Így oldalbetöltés nélkül változtathatja a felhasználó a jelentés típusát.

# A NewReportSecondView nézet

A *NewReportFirstView Next* gombjának megnyomásával jut ide a felhasználó.

Ez a nézet csak egy ellenőrző felületet biztosít a felhasználónak az előző nézettben megadott értékekkel. Itt tud menteni a *Save* gomb megnyomásával, vagy léphet vissza az előző nézetbe az *Edit* gomb megnyomásával, ha valamit változtatni szeretne a jelentésén. Az C*ancel* gombot megnyomva pedig újraindítja a *New report* funkciót.

# A ReportTableView nézet

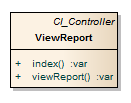
Egy olyan nézet, amely két funkcióból is elérhető. Ezekből az egyik a *New report*. Egy táblázatban jeleníti meg egy adott hónapra levetítve felhasználó által bevitt jelentéseket.

Egy legördülő lista segítségével választhat hónapot a felhasználó. A *New report* funkciónál a lista alapértelmezett értéke az a hónap, amire a jelentésünk szólt.

# A View report oldal

Ezen az oldalon tudjuk megtekinteni a bevitt jelentéseinket.

# A ViewReport vezérlő



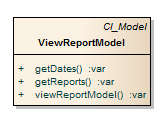
*A ViewReport osztálydiagramja*

A *View report* funkció vezérlő osztálya.

A következő eljárásai vannak:

* **public function** newReport(): a konstruktor betölti a *NewReportModel* és a *LoginModel* modelleket;
* **public function** index(): meghívja a *ViewReportModel* modell *getReports()* és *getDates()* függvényeit, visszatérési értéküket eltárolja a *data* tömbben, és betölti a *ViewReportView* nézetet.

# A ViewReportModel modell



*A ViewReportModel osztálydiagramja*

Az *View report* funkciómodell osztálya. Függvényei:

* **public function** getReports(): visszaadja a *team member* jelentéseit;
* **public function** getDates($value): visszaadja azokat a hónapokat, amelyekhez létezik a *team member*nek jelentése.

# A ViewReportView nézet

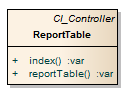
A *View report* nézete.

Egy úgy jeleníti meg a jelentéseket, mint a *NewReportSecondView* nézet, de itt van két navigációs nyíl is a jelentés két oldalán, illetve fölötte egy legördülő lista, amiben a hónapot választhatjuk ki. A navigációért JavaScript függvények felelnek.

# A Report table oldal

Ez az oldal egy táblázatban foglalja össze a *team member* felhasználó jelentéseit.

# A ReportTable vezérlő



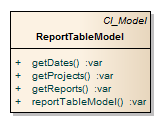
*A ReportTable osztálydiagramja*

A *Report table* funkció vezérlő osztálya.

Egy eljárása van:

* **public function** index(): meghívja a *ReportTableModel* modell *getReports(),* *getProjects()* és *getDates()* függvényeit, visszatérési értéküket változókban tárolja el. A *message* és a *month* változókat „*Report table*” értékkel tölti fel. Végül betölti a *ReportTableView* nézetet.

# A ReportTableModel modell



*A ReportTableModel osztálydiagramja*

Az *Report table* funkciómodell osztálya. Függvényei:

* **public function** getReports(): visszaadja a *team member* jelentéseit;
* **public function** getProjects(): visszaadja a project tábla sorait;
* **public function** getDates(): visszaadja azokat a hónapokat, amelyekhez létezik a *team member*nek jelentése.

# A ReportTableView nézet

A *Report table* nézete.

Egy táblázatot tartalmaz, amiben a jelentések összesítései szerepelnek a *team member* jelentéseinek adataival. Egy legördülő listában lehet kiválasztani az adott hónapot.

# A Help oldal

A *Help* oldalait egy-egy vezérlő és egy-egy nézet alkotja, a működésüket annak egyszerűsége miatt nem fejtem ki.

# Az executive jogú felhasználók által elérhető funkciók

Ezen oldalak csak akkor érhetőek el egy felhasználó számára, ha a szerepe és a szellemszerepe is *executive*.

# A New Report oldal

Osztályainak, valamint a nézeteinek a szerkezete megegyezik a *team member* által elérhető funkcióval, az egyetlen különbség, hogy itt a *Team member name* bemeneti érték is egy legördülő listából választható ki.

# A View Report oldal

Osztályainak, valamint a nézeteinek a szerkezete megegyezik a *team member* által elérhető funkcióval, az egyetlen különbség, hogy itt a *Team member name* bemeneti érték is egy legördülő listából választható ki.

# A Report table oldal

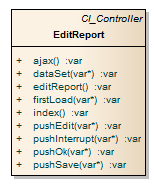
Osztályainak, valamint a nézeteinek a szerkezete megegyezik a *team member* által elérhető funkcióval, az egyetlen különbség, hogy itt a *Team member* name bemeneti érték is egy legördülő listából választható ki.

# Az Edit report oldal

Ez az oldal ad lehetőséget az executive jogú felhasználóknak a jelentések módosítására.

# Az EditReport vezérlő

Az *Edit report* funkció vezérlő osztálya.

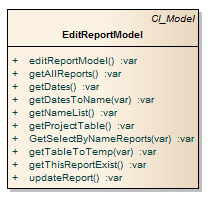


*Az EditReport osztálydiagramja*

A következő eljárásai vannak:

* **public function** editReport(): a konstruktor betölti a *NewReportModel* és a LoginModel modelleket;
* **public function** index(): meghívja a *dataSet()* eljárást, ami a *data* tömb kezdeti értékét állítja be. Majd meghívja a *firstLoad()* eljárást, ami a lényegében az oldal inicializálásáért felelős. A funkció nézeteinek különböző gombjainak hatására meghívja a szükséges segédeljárásait. Betölti az *EditReportFrameView* nézetet, ami pedig a *change\_view* változóban tárolt értéknek megfelelő nézetet tölti be. A változó értéke kezdetben 0;
* **public function** dataSet(&$data): a $data tömb kezdeti értékeit állítja be. Ehhez meghívja az *EditReportModel* szükséges függvényeit és beolvassa az *EditReportFirstView* nézet bemeneti mezőinek értékeit. Ennek a változónak a segítségével állítja be a nézet a legördülő listáinak kezdeti értékeit;
* **public function** pushOk(&$data): az *EditReportSecondView* nézet *Next* gombjának megnyomására hívja meg az *index()* eljárás. Az eljárás ellenőrzi, hogy a jelentés létezik-e már, a munkaidő nem 0-e, illetve a jelentés kezdeti és végdátuma egy hónapba esik-e. Ha minden megfelelő, akkor jelentés típusát beállítja a felhasználó által kijelölt rádiógombnak megfelelően. A *change\_view* változó értékét 2-re állítja, aminek hatására betöltődik az *EditReportThirdView* nézet;
* **public function** pushSave(&$data): az *EditReportThirdView* nézet *Save* gombjának megnyomásakor hívja meg az *index()* eljárás. Ellenőrzi, hogy nem az *Ok* gomb megnyomásával és böngészőkbe ágyazott vissza gomb segítségével jutott ide a felhasználó, így kizárja a lehetőségét annak, hogy újra elmenthessük már meglévő jelentésünket. Ha a felhasználó mégis ezzel próbálkozik, a *change\_view* értékét 1-ra állítja, visszatölti az *EditReportSecondView* nézetet és hibaüzenetben figyelmezteti a felhasználót: „The Job report exist!”. Ha minden rendeltetésszerűen zajlik, meghívja az *EditReportModel saveReport()* függvényét ezzel elmentve az adatbázisba a jelentést. Ezután betölti a *ReportTableModel* modellt. Meghívja a *getReports()* és a *getProjects()* függvényeit és elmenti ezeket egy-egy változóba. A *change\_view* változót 3-ra állítja ezzel betölti *ReportTableView* nézetet, ami a jelentéseknek egy táblázata;
* **public function** pushEdit(&$data): az *EditReportFirstView* vagy az *EditReportThirdView* nézet *Edit* gombjának megnyomásakor hívja meg az index() eljárás. Elmenti a jelentés típusát, és a *change\_view* változót 1-ra állítja. Így újra az *EditReportSecondView* nézetet tölti be a felhasználónak, így módosíthatja a jelentését;
* **public function** pushInterrupt(&$data): az EditReportThirdView nézet *Cancel* gombjának megnyomásakor hívja meg az *index()* eljárás. Megszakítja a jelentés bevitelét, betölti az *EditReportFirstView* nézetet annak inicializáláskori értékeivel;
* firstLoad(&$data): az első oldalbetöltésnél hívja meg az *index()* eljárás. Meghívja az *EditReportModel* modell *getNameList()* és *getAllReports()* függvényét, amelyek visszaadják a *name* tábla sorait, illetve a jelentések sorait;
* **public function** ajax(): betölti az *EditReportModel* modellt, majd az *EditReportFirstView* Ajax hívását továbbítja a neki. Ezután elvégez egy vizsgálatot a nézettől átvett paraméter értékén. A paraméter értékében egy *team member* nevének azonosítója vagy az "*all"* érték van tárolva, ami alapján meghívja a modell megfelelő függvényét, ami a hívás paraméterének értéke alapján adja vissza a jelentéseket.

# Az EditReportModel modell



*Az EditReportModel osztálydiagramja*

Az New report funkciómodell osztálya. Függvényei:

* **public function** getProjectTable(): visszaadja a project tábla sorait;
* **public function** getTableToTemp($value): visszaadja a paraméterének megfelelő tábla sorait;
* **public function** getThisReportExist(): visszaadja, hogy a felhasználó által megváltoztatott jelentés létezik-e;
* **public function** updateReport(): menti a felhasználó által bevitt változtatásokat a jelentésben;
* **public function** getAllReports(): visszaadja az összes jelentést;
* **public function** getNameList(): a nevek táblát adja vissza;
* **public function** getSelectByNameReports($name\_id): egy felhasználói azonosítóhoz adja vissza az összes jelentést;
* **public function** getDates(): rendezetten visszaadja azokat a hónapokat, amelyekhez létezik mentett jelentés;
* **public function** getDatesToName($name\_id): rendezetten visszaadja azokat a hónapokat, amelyekhez létezik olyan mentett jelentés, amelyiknek a felhasználói azonosítója megegyezik a paraméterben kapottal.

# Az EditReportFrameView nézet

A központi nézet, ez tölti be a többi nézetet a *change\_view* változó értékének megfelelően.

# Az EditReportFirstView nézet

Az *Edit report* kezdeti nézete.

A nézet a szokásos táblázatban tartalmazza az aktuális jelentés értékeit.

Ez a nézet fogadja a felhasználót, ahol egy-egy legördülő listából választhatja ki a *team member* nevét, akinek a jelentésére kíváncsi, és a hónapot.

Ezt követően nyilak segítségével navigálhat a jelentések között. Ezt JavaScript segítségével valósítottam meg, így oldalbetöltés nélkül működik a funkció.

# Az EditReportSecondView nézet

Az *EditReportFirstView Edit* gombjának megnyomásával jut ide a felhasználó.

Ez a nézet a funkció editálásért felelős nézete. Táblázatban tartalmazza a korábbi nézetben kiválasztott jelentés adatait, amelyeket legördülő listák és bemeneti mezők segítségével változtathat a felhasználó. Rádiógombok segítségével változtathat a jelentés típusán is. A típusváltást JavaScript vezérli, ezért itt sem történik oldalbetöltés közben.

# Az EditReportThirdView nézet

Az *EditReportSecondView Next* gombjának megnyomásával jut ide a felhasználó.

Ez a nézet a funkció editálásért felelős nézete. Ez egy ellenőrző nézet, ahol még egy utolsó pillantást vethetünk a megváltoztatni kívánt jelentésünk leendő adataira. *Save,* *Edit* és *Cancel* gombokkal indíthatunk felhasználói akciót. A *Save* megnyomásának hatására mentjük a jelentésünket és betöltjük a *ReportTableView* oldalt. Az *Edit* hatására visszajutunk az előző nézetre, a *Cancel* pedig megszakítja a műveletet és újra indítja az *Edit report* funkciót.

# A ReportTableView nézet

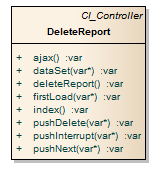
Egy olyan nézet, amely több funkcióból is elérhető. Ezekből az egyik az *Edit report*. Egy táblázatban jeleníti meg egy adott hónapra levetítve a felhasználó által bevitt jelentéseket.

Egy legördülő lista segítségével választhat hónapot a felhasználó. Az *Edit report* funkciónál a lista alapértelmezett értéke az a hónap, amire a jelentésünk szólt.

# A Delete report oldal

Ez az oldal ad lehetőséget az *executive* jogú felhasználóknak a jelentések törlésére.

# A DeleteReport vezérlő



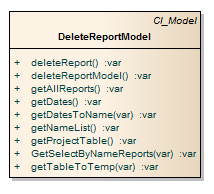
*A DeleteReport osztálydiagramja*

Az *Delete report* funkció vezérlő osztálya.

A következő eljárásai vannak:

* **public function** deleteReport(): a konstruktor betölti a *DeleteReportModel* és a *LoginModel* modelleket;
* **public function** Index(): meghívja a *dataSet()* eljárást, ami a *data* tömb kezdeti értékét állítja be. Majd meghívja a *firstLoad()* eljárást, ami a lényegében az oldal inicializálásáért felelős. A funkció nézeteinek különböző gombjainak hatására, megnyomására meghívja a szükséges segédeljárásait. Betölti a *DeleteReportFrameView* nézetet, ami pedig a *change\_view* változóban tárolt értéknek megfelelő nézetet tölti be. A változó értéke kezdetben 0;
* **public function** dataSet(&$data): a *data* tömb kezdeti értékeit állítja be. Ehhez meghívja a *DeleteReportModel* szükséges függvényeit és beolvassa a *DeleteReportSecondView* nézet bemeneti mezőinek értékeit. Ennek a változónak a segítségével állítja be a nézet a legördülő listáinak kezdeti értékeit;
* **public function** pushNext(&$data): a *DeleteReportFirstView* nézet *Next* gombjának megnyomására hívja meg az *index()* eljárás. Az eljárás a jelentés típusát beállítja a felhasználó által kijelölt rádiógombnak megfelelően. A *change\_view* változó értékét 1-re állítja, aminek hatására betöltődik a *DeleteReportSecondView* nézet;
* **public function** pushDelete(&$data): a *DeleteReportSecondView* nézet *Delete* gombjának megnyomásakor hívja meg az index() eljárás. Meghívja a *DeleteReportModel deleteReport()* függvényét ezzel átállítva a jelentés *validity* jellemzőjét hamisra. Ezután meghívja a *DeleteReportModel getAllReports(),* *getDates()* és *getNameList()* függvényeit és elmenti ezeket egy-egy változóba. A *change\_view* változót 0-ra állítja, ezzel betölti *DeleteReportFirstView* nézetet;
* **public function** pushInterrupt(&$data): a *DeleteReportSecondView* nézet *Cancel* gombjának megnyomásakor hívja meg az *index()* eljárás. Megszakítja a jelentés törlését, betölti az *DeleteReportFirstView* nézetet annak inicializáláskori értékeivel;
* **public function** firstLoad(&$data): az első oldalbetöltésnél hívja meg az *index()* eljárás. Meghívja a *DeleteReportModel getAllReports(), getDates()* és *getNameList()* függvényeit és elmenti ezeket egy-egy változóba. A *change\_view* változót 0-ra állítja, ezzel betölti *DeleteReportFirstView* nézetet;
* **public function** ajax(): betölti a *DeleteReportModel* modellt, majd az *DeleteReportFirstView* Ajax hívását továbbítja a neki. Ezután elvégez egy vizsgálatot a nézettől átvett paraméter értékén. A paraméter értékében egy *team member* nevének azonosítója vagy az „*all*” érték van tárolva, ami alapján meghívja a modell megfelelő függvényét, ami a hívás paraméterének értéke alapján adja vissza a jelentéseket.

# A DeleteReportModel modell



*A DeleteReportModel osztálydiagramja*

Az *Delete report* funkciómodell osztálya. Függvényei:

* **public function** getProjectTable(): visszaadja a project tábla sorait;
* **public function** getTableToTemp($value): visszaadja a paraméterének megfelelő tábla sorait;
* **public function** getAllReports(): visszaadja az összes jelentést;
* **public function** getNameList(): a nevek táblát adja vissza;
* **public function** getSelectByNameReports($name\_id): egy felhasználói azonosítóhoz adja vissza az összes jelentést;
* **public function** getDates(): rendezetten visszaadja azokat a hónapokat, amelyekhez létezik mentett jelentés;
* **public function** getDatesToName($name\_id): rendezetten visszaadja azokat a hónapokat, amelyekhez létezik olyan mentett jelentés, amelyiknek a felhasználói azonosítója megegyezik a paraméterben kapottal;
* **public function** deleteReport(): a felhasználó által bevitt jelentés *validity* attribútumát hamisra állítja.

# A DeleteReportFrameView nézet

A központi nézet, ez tölti be a többi nézetet a *change\_view* változó értékének megfelelően.

# A DeleteReportFirstView nézet

A *Delete report* kezdeti nézete.

A nézet a szokásos táblázatban tartalmazza az aktuális jelentés értékeit.

Ez a nézet fogadja a felhasználót, ahol egy-egy legördülő listából választhatja ki a *team member* nevét, akinek a jelentésére kíváncsi, és a hónapot.

Ezt követően nyilak segítségével navigálhat a jelentések között. Ezt JavaScript segítségével valósítottam meg, így oldalbetöltés nélkül működik a funkció.

# A DeleteReportSecondView nézet

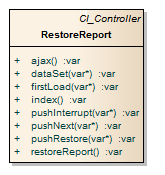
A *DeleteReportFirtsView Next* gombjának megnyomásával jut ide a felhasználó.

Ez egy ellenőrző nézet, ahol még egy utolsó pillantást vethetünk a törölni kívánt jelentésünk leendő adataira. *Delete* és *Cancel* gombokkal indíthatunk felhasználói akciót. A *Delete* megnyomásának hatására töröljük a jelentésünket, azaz a *validity* attribútumát hamisra állítjuk, és visszatöltjük *DeleteReportFirtsView* oldalt. A *Cancel* pedig megszakítja a műveletet és újraindítja a *Delete report* funkciót.

# A Restore report oldal

A *Delete report* inverz funkciója.

# A RestoreReport vezérlő



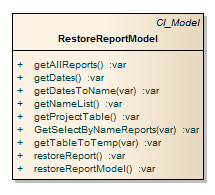
*A RestoreReport osztálydiagramja*

Az *Restore report* funkció vezérlő osztálya.

A következő eljárásai vannak:

* **public function** RestoreReport(): a konstruktor betölti a *RestoreReportModel* és a *LoginModel* modelleket;
* **public function** index(): meghívja a *dataSet()* eljárást, ami a *data* tömb kezdeti értékét állítja be. Majd meghívja a *firstLoad()* eljárást, ami a lényegében az oldal inicializálásáért felelős. A funkció nézeteinek különböző gombjainak hatására meghívja a szükséges segédeljárásait. Betölti a *RestoreReportFrameView* nézetet, ami pedig a *change\_view* változóban tárolt értéknek megfelelő nézetet tölti be. A változó értéke kezdetben 0;
* **public function** dataSet(&$data): a *data* tömb kezdeti értékeit állítja be. Ehhez meghívja a *RestoreReportModel* szükséges függvényeit és beolvassa a *RestoreReportSecondView* nézet bemeneti mezőinek értékeit;
* **public function** pushNext(&$data): a *RestoreReportFirstView* nézet *Next* gombjának megnyomására hívja meg az *index()* eljárás. A *change\_view* változó értékét 1-re állítja, aminek hatására betöltődik a *RestoreReportSecondView* nézet;
* **public function** pushRestore(&$data): a *RestoreReportSecondView* nézet *Restore* gombjának megnyomásakor hívja meg az *index()* eljárás. Meghívja a *RestoreReportModel restoreReport()* függvényét ezzel átállítva a jelentés *validity* jellemzőjét igazra. Ezután meghívja a *RestoreReportModel getAllReports(), getDates() és getNameList()* függvényeit és elmenti ezeket egy-egy változóba. A *change\_view* változót 0-ra állítja, ezzel betölti *RestoreReportFirstView* nézetet;
* **public function** pushInterrupt(&$data): a *RestoreReportSecondView* nézet *Cancel* gombjának megnyomásakor hívja meg *az index()* eljárás. Megszakítja a jelentés törlését, betölti a *RestoreReportFirstView* nézetet annak inicializáláskori értékeivel;
* **public function** firstLoad(&$data): az első oldalbetöltésnél hívja meg az *index()* eljárás. Meghívja a *RestoreReportModel getAllReports(), getDates()* és *getNameList()* függvényeit és elmenti ezeket egy-egy változóba. A *change\_view* változót 0-ra állítja, ezzel betölti *RestoreReportFirstView* nézetet;
* **public function** ajax(): betölti a *RestoreReportModel* modellt, majd az *RestoreReportFirstView* Ajax hívását továbbítja a neki. Ezután elvégez egy vizsgálatot a nézettől átvett paraméter értékén. A paraméter értékében egy *team member* nevének azonosítója vagy az „*all*” érték van tárolva, ami alapján meghívja a modell megfelelő függvényét, ami a hívás paraméterének értéke alapján adja vissza a jelentéseket.

# A RestoreReportModel modell



*A RestoreReportModel osztálydiagramja*

Az *Restore report* funkciómodell osztálya. Függvényei:

* **public function** getProjectTable(): visszaadja a *project* tábla sorait;
* **public function** getTableToTemp($value): visszaadja a paraméterének megfelelő tábla sorait;
* **public function** getAllReports(): visszaadja az összes jelentést;
* **public function** getNameList(): a *name* táblát adja vissza;
* **public function** getSelectByNameReports($name\_id): egy felhasználói azonosítóhoz adja vissza az összes jelentést;
* **public function** getDates(): rendezetten visszaadja azokat a hónapokat, amelyekhez létezik mentett jelentés;
* **public function** getDatesToName($name\_id): rendezetten visszaadja azokat a hónapokat, amelyekhez létezik olyan mentett jelentés, amelyiknek a felhasználói azonosítója megegyezik a paraméterben kapottal;
* **public function** restoreReport(): a felhasználó által bevitt jelentés *validity* attribútumát igazra állítja.

# A RestoreReportFrameView nézet

A központi nézet, ez tölti be a többi nézetet a *change\_view* változó értékének megfelelően.

# A RestoreReportFirstView nézet

A *Restore report* kezdeti nézete.

A nézet a szokásos táblázatban tartalmazza az aktuális jelentés értékeit.

Ez a nézet fogadja a felhasználót, ahol egy-egy legördülő listából választhatja ki a *team member* nevét, akinek a jelentésére kíváncsi, és a hónapot.

Ezt követően nyilak segítségével navigálhat a jelentések között. Ezt JavaScript segítségével valósítottam meg, így oldalbetöltés nélkül működik a funkció.

# A RestoreReportSecondView nézet

Az *RestoreReportFirtsView Next* gombjának megnyomásával jut ide a felhasználó.

Ez egy ellenőrző nézet, ahol még egy utolsó pillantást vethetünk a törlésből visszaállítani kívánt jelentésünk leendő adataira. *Restore* és *Cancel* gombokkal indíthatunk felhasználói akciót. A Restore megnyomásának hatására visszaállítjuk a jelentésünket, azaz a *validity* attribútumát igazra állítjuk, és visszatöltjük *RestoreReportFirtsView* oldalt. A *Cancel* pedig megszakítja a műveletet és újra indítja a *Restore report* funkciót.

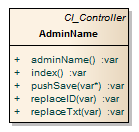
# Adminisztráció

Ez a funkció felületet nyújt az adatbázis feltöltésére és karbantartására.

# A felhasználók kezelése

Ezen az oldalon tud egy *executive* jogú felhasználó felhasználókat törölni és hozzáadni a rendszerhez.

# Az AdminName vezérlő



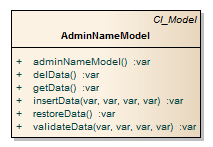
*Az AdminName osztálydiagramja*

A falhasználói adminisztrációt megvalósító vezérlő osztálya.

A következő függvényei és eljárásai vannak:

* **public function** index(): meghívja az *AdminNameModel* modell *getData()* függvényét, ami a felhasználókat adja vissza. Betölti az *AdminNameView* nézetet. A nézet gombjainak megfelelően meghívja a modell megfelelő függvényeit, és szükség esetén frissíti a felhasználókat tároló változót;
* **public function** replaceTxt( $a ): ez a függvény visszaad egy karaktersorozatot, levágva szóközöket a paraméterként megadott karaktersorozat elejéről és végéről, valamint az egymás melletti szóközöket. A függvény által visszaadott értéket mentjük az adatbázisba;
* **public function** replaceID ( $a ): Levágja a hosszú szóközöket a paraméterként megadott karaktersorozatról;
* **public function** pushSave(&$data): Az *AdminNameView* nézet *Save* gombjának megnyomásakor hívja meg az index() eljárás. Ellenőrzi, hogy létezik-e a felhasználó, akit szeretnénk menteni. Ha létezik és nincs törölt státuszban figyelmeztetést küld a nézeten keresztül. Ha törölt státuszban van, felülírja az attribútumait, ha pedig nincs az adatbázisban, akkor létrehoz egy új sort a *name* táblában a felhasználó által megadott értékekkel.

# Az AdminNameModel modell



*Az AdminNameModel osztálydiagramja*

A felhasználói adminisztráció funkciómodell osztálya. Függvényei:

* **public function** getData(): visszaadja a *name* tábla sorait;
* **public function** insertData($username,$name,$password,$role): menti az új felhasználót a *name* táblába;
* **public function** validateData($username,$name,$password,$role): ha már létezik az adott felhasználó, de törölt státuszban van, felülírja a paraméterként kapott értékekkel;
* **public function** delData(): az adott felhasználóhoz tartozó sorban a *validity* attribútum értékét hamisra állítja;
* **public function** restoreData(): az adott felhasználóhoz tartozó sorban a *validity* attribútum értékét igazra állítja.

# Az AdminNameView nézet

Bemeneti mezőket tartalmaz a felvinni kívánt felhasználó felhasználónevének, nevének és jelszavának. Egy legördülő listában választhatjuk ki a felhasználó típusát: "*Executive"* vagy "*Team member"*. A *Save* gombbal tudjuk menteni. További nem szerkeszthető mezőket tartalmaz az oldal, amelyek az aktív és passzív felhasználók adatait jelenítik meg, melyeket szintén egy-egy legördülő listában választhatunk ki.

A további adminisztrációs funkciók megvalósítása minden esetben megegyezik az előbb vázolttal, ezért ezek kifejtésére nem térek ki. A funkciók a következő táblákat töltik fel adatokkal:

* *project*
* *job*
* *activity*
* *product*
* *comp\_name*
* *comp\_version*
* *story\_title*
* *story\_pos*
* *work\_title*
* *work\_pos*
* *lifecycle*
* *lifecycle\_act*

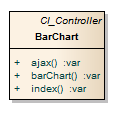
# A Monitoring funkció

A grafikus adatmegjelenítésért felelős oldalak tartoznak ide.

# A Bar Chart oldal

Egy sávos szerkezetű grafikus adatmegjelenítő oldal.

# A BarChart vezérlő



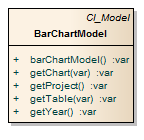
*A BarChart osztálydiagramja*

A *Bar Chart* grafikus adatmegjelenítő funkció vezérlő osztálya.

A következő függvényei és eljárásai vannak:

* **public function** index(): meghívja a *PieChartModel* modell *getYear()* függvényét, ami azokat az éveket adja vissza, amelyekben történt jelentés bevitel. Meghívja a modell *getProject()* függvényét is, amely a projekt tábla sorait adja vissza. Majd meghívja a *getTable($value)* függvényt a következő paraméterekkel: *name, job, activity, product, comp\_name, comp\_version, story\_title, story\_pos, work\_title, work\_pos, lifecycle, lifecycle\_act*. Ez nem csinál mást, mint visszaadja a paraméteréül kapott tábla sorait. A meghívott függvények visszatérési értékeit változókban elmenti, majd betölti a *PieChartView* nézetet átadva neki a változókat;
* **public function** ajax(): betölti a *PieChartModel* modellt, majd az *PieChartView* nézet Ajax hívását továbbítja neki. A nézet Ajax hívása egy évszámot tartalmaz, amivel meghívja a modell *getChart($year)* függvényét, az pedig visszaadja az adott évhez tartozó jelentéseket csoportosítva év, hónap és a *job\_report* tábla egyes attribútumai szerint. Végül a modelltől kapott tömböt továbbítja a nézet felé, ami majd ez alapján építi fel a grafikont.

# A BarChartModel modell



*A BarChartModel osztálydiagramja*

A *Bar Chart* grafikus adatmegjelenítő funkciómodell osztálya. Függvényei:

* **public function** getChart($year): visszaadja a *job\_report* tábla azon sorait, melyek a paraméterül kapott év valamely hónapjának időszakára lettek elmentve, csoportosítva év, hónap és a *job\_report* tábla egyes attribútumai szerint;
* **public function** getTable($value): visszaadja a paraméteréül kapott tábla sorait.
* **public function** getProject(): visszaadja a *project* tábla sorait;
* **public function** getYear(): azokat az éveket adja vissza, amelyekben történt jelentés bevitel.

# A BarChartView nézet

A nézet alapját a Highcharts JavaScript könyvtár egy grafikonja adja, amely hónapokra lebontva, sávokban jeleníti meg a felhasználó által kiválasztott adatokat. A nézet fölső felén *Chart Options* gomb található, amelyet megnyomva megjelenik, illetve eltűnik a grafikon kezelőfelülete. Kezdetben a kezelőfelület rejtve van. Ez a kezelőfelület lényegében egy általam átalakított JQuery vezérlő, amely 14 fület tartalmaz: *Date, projekt, Team member, Job, Activity, Product, Component name, Component version, Story title, Story position, Work title, Work position, Lifecylce, Lifecycle activity*.

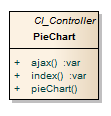
Ebből a *Date* fül rádiógombokat tartalmaz a kiválasztható évszámokkal, a többi fül pedig jelölőnégyzeteket tartalmaz a hozzájuk tartozó tábla sorainak hasonló nevű attribútumainak értékeivel. A jelölőnégyzeteket tartalmazó füleknél található még 2 gomb, amellyel az összes a fülhöz tartozó jelölődobozt aktiválhatjuk, vagy deaktiválhatjuk. A grafikus megjelenítés dinamikusan zajlik, amelyet JavaScript irányít. Minden felhasználó akciót követően végigmegy a vezérlőtől Ajax hívás útján kapott jelentéseken, és amelyhez minden fülben talál kijelölt elemet azt hozzáadja a megfelelő hónaphoz. Az Ajax hívást egy adott évszám rádiógombjának a kiválasztása indítja el. A nézet tetején egy link található *Manpower pie* felirattal, mely egy másik grafikus adatmegjelenítő funkcióhoz vezérli a felhasználót.

A nézet alján további két link található, amelyek a jelentéskészítéshez, illetve a *Monitoring* táblázataihoz vezetnek.

# A Pie Chart oldal

Ez az oldal az adatbázis adatait egy kördiagramban jeleníti meg.

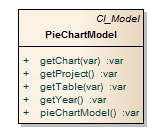
# A PieChart vezérlő



*A PieChart osztálydiagramja*

A *Pie Chart* grafikus adatmegjelenítő funkció vezérlő osztálya.

# A PieChartModel modell



*A PieChartModel osztálydiagramja*

A *Pie Chart* grafikus adatmegjelenítő funkciómodell osztálya. A modell szerkezete teljesen megegyezik a *PieChartModel* modell szerkezetével.

# A PieChartView nézet

A nézet szerkezete teljesen megegyezik a *BarChartView* nézet szerkezetével. A különbség köztük abban van, hogy ez egy kördiagramon alapul.

# Monitoring táblázatok

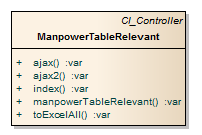
A monitoring menüpont a *Manpower tables* almenüpontjából érhető el. Ez a funkció egy táblázatban megjeleníti a munkatársak jelentéseit. Sorait a project tábla project attribútumának értékei. Ezen kívül van még két összegző sor is. Oszlopai pedig a munkatársak. Két nagyon hasonló oldal tartozik ehhez a funkcióhoz:

* *ManpowerTableAll;*
* *ManpowerTableRelevant.*

# A ManpowerTableRelevant oldal

Ez z oldal csak azokat a munkatársakat jeleníti meg a táblázatban, akik készítettek jelentést.

# A ManpowerTableRelevant vezérlő



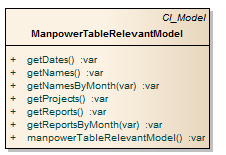
*A ManpowerTableRelevant osztálydiagramja*

Ez az osztály a vele azonos nevű oldal vezérlője.

A következő függvényei és eljárásai vannak:

* **public function** index(): meghívja a *ManpowerRelevantModel* *getDates()*, *getNames()* és *getProjects()* függvényeit, eltárolja ezeket egy változóban és betölti a *ManpowerTableRelevantView* nézetet. Amennyiben a felhasználó megnyomja a nézet *Excel export* gombját, meghívja a *toExcelAll()* eljárást;
* **public function** toExcelAll(): ez az eljárás vezérli a táblázat Excel formátumba történő exportálását. A *ManpowerRelevantModel* függvényeitől visszakapott értékeket továbbítja *ManpowerTableRelevantToXls* nézetnek, ami az Excel generálást végzi;
* **public function** ajax(): a *ManpowerTableRelevantView* nézet egyik Ajax hívása;
* **public function** ajax2(): a *ManpowerTableRelevantView* nézet másik Ajax hívása.

# A ManpowerTableRelevantModel modell



*A ManpowerTableRelevantModel osztálydiagramja*

A *ManpowerTableRelevant* oldalmodell osztálya. Függvényei:

* **public function** getDates(): visszaadja a *job\_report* tábla azon hónapjait, amikor történt jelentés bevitel az adatbázisba;
* **public function** getReports(): visszaadja a jelentések és a rájuk jelentett munkaidők összesítéseit, *project\_id* és felhasználónév szerint csoportosítva;
* **public function** getNames(): visszaadja a nem törölt felhasználókat;
* **public function** getNamesByMonth($dates): visszaadja azokat a neveket, amelyekkel a paraméterül kapott hónapban történt jelentés;
* **public function** getProjects(): visszaadja a project tábla sorait;
* **public function** getReportByMonth($dates): visszaadja paraméterül kapott hónapban történt jelentéseket;

# A ManpowerTableRelevant nézetei

A funkciónak két nézete van. Az egyik az adott oldal nézete, a másik az Excel exportért felel.

1. **A ManpowerTableRelevantView nézet**

Ez a nézet egy táblázatban összesíti a munkaidő jelentéseket. Sorait a *projec*t tábla *project* attribútumának értékei, és azok összegei adják. Oszlopai pedig azon munkatársak, akik készítettek jelentést.

1. **A ManpowerTableRelevantToXls nézet**

Ez a nézet felelős az Excel táblázatba történő exportért. Ehhez előre megírt PHP könyvtárakat használok. Az előző nézet táblázatát exportálja egy Excel fájlba.

# A ManpowerTableAll oldal

Ez az oldal teljesen megegyezik a *ManpowerTableRelevant* oldallal, azzal a különbséggel, hogy minden munkatársat megjelenít a táblázat oszlopában. Az osztályai és a nézetei viszont annyira hasonlóak, hogy azok részletes bemutatásától eltekintek.

# Tesztelés

A program tesztelése két fázisból állt. Először az egyes osztályokon, illetve oldalakon végeztem modultesztet. Ezután a kapcsolódó oldalak integrációs tesztje következett, melynek utolsó fázisa a teljes program működésének tesztelése volt.

# Modultesztelés

A modultesztelés során a fő cél az egyes modulok önálló működési hibáinak detektálása volt. Ezen tesztek során felszínre kerültek adatszerkezeti hibák, az adatbázis elérési és konzisztenciával kapcsolatos problémák. Ezeket minden esetben javítottam.

# Integrációs tesztek

Az integrációs teszt alatt az osztályok együttes működését ellenőriztem. Amennyiben hiba merült fel, úgy az abban szereplő osztályokat illetve megfelelő metódusaikat újra modultesztelés alá vetettem. Az integrációs teszt utolsó szakaszában a teljes programot vetettem funkcionális teszt alá.

# ÖSSZEFOGLALÁS

Egy munkatevékenység és munkaidő jelentés készítésére, kezelésére, továbbá a munkafolyamatok követésére és irányítására alkalmas rendszer megvalósítása volt a feladatom a szakdolgozatom során.

Munkámat a követelmények felvázolásával kezdtem. A fejlesztőeszközök kiválasztását követően az adatbázist úgy terveztem meg, hogy eleget tegyen a követelményeknek, majd megvalósítottam azokat a funkciókat, amelyek a jelentések beviteléhez és kezeléséhez szükségesek. Ezt követte a különböző grafikai és táblázatos megjelenítők implementációja. Az autentikáció megvalósítása után pedig elkészítettem az adminisztrációs felületet.

A tesztelés alatt bebizonyosodott, hogy a program stabil és eleget teszt a követelményeknek.

Az általam választott fejlesztőeszközök alkalmasnak bizonyultak az alkalmazás implementálására.

A szoftver segítségével a fejlesztés munkatársai el tudják készíteni a jelentéseiket, a fejlesztés vezetője nyomon tudja követni a munkafolyamatokat, be tud avatkozni, ha az szükséges. A program továbbá felépítésénél fogva rugalmas, alkalmas a jövőbeni bővítésekre ugyanúgy, mint a változtatásokra. A fejlesztési folyamat elérte a célját, ezért sikeresnek minősíthető.

# KÖSZÖNETNYILVÁNÍTÁS

Szeretném hálámat kifejezni mindazoknak, akik valamilyen módon hozzájárultak a szakdolgozatom elkészültéhez.

Köszönet illeti főnökömet és minden kollegámat, akiktől tanulhattam.

Hálámat szeretném kifejezni továbbá Szentes Gábornak, aki felbecsülhetetlen értékű segítséget nyújtott egyebek mellett a dolgozatban használt keretrendszer megismerésében.

Külön köszönetemet szeretném kifejezni Legéndi Richárd Olivérnek, aki részletes, de lényegre törő tanácsaival, építő jellegű kritikáival állandó fejlődésre ösztönzött.

Végül, de nem utolsósorban egyetemi témafelelősömnek, Török Márknak szeretném megköszönni azt a segítséget, amit a szakdolgozatom elkészítéséhez nyújtott.

# IRODALOMJEGYZÉK

1. Apache [Online]. Available: http://httpd.apache.org/docs/2.2/

[Hozzáférés dátuma: 04 2011]

1. MySQL [Online]. Available: http://dev.mysql.com/doc/

[Hozzáférés dátuma: 04 2011]

1. PHP [Online]. Available: http://www.php.net/manual/en/

[Hozzáférés dátuma: 04 2011]

1. EasyPHP [Online]. Available: http://www.easyphp.org/

[Hozzáférés dátuma: 04 2011]

1. w3schools.com [Online]. Available: http://www.w3schools.com/

[Hozzáférés dátuma: 04 2011]

1. JQuery [Online]. Available: http://docs.jquery.com/Main\_Page

[Hozzáférés dátuma: 04 2011]

1. JQueryUI [Online]. Available: http://jqueryui.com/demos/

[Hozzáférés dátuma: 04 2011]

1. Highcharts [Online]. Available: http://www.highcharts.com/

[Hozzáférés dátuma: 04 2011]

1. Codeigniter [Online]. Available: http://codeigniter.com/user\_guide/

[Hozzáférés dátuma: 04 2011]

# CD MELLÉKLET

**A mellékelt CD tartalma**

A CD-n megtalálható a program futtatásához szükséges EasyPHP installáló fájlja, a Codeigniter keretrendszer, a program forráskódja, illetve ez a dokumentáció.

Információk az Mx program használatához:

1. Amennyiben számítógépén nincs telepítetve az EasyPHP, installálja azt számítógépére a CD EasyPHP könyvtárából;
2. A program használatához másolja az Mx könyvtárat a telepített webszerver gyökérkönyvtárába és indítson el egy böngészőt. Az Mx könyvtár tartalmazza a Codeigniter keretrendszert is;
3. Az adatbázis tartalmaz példaadatokat, ezeket módosíthatja, törölheti, ld. bővebben a felhasználói dokumentációt;
4. Felhasználóként való belépéshez: Felhasználó név: gipsjahu; Jelszó: gipsjahu (kezdeti beállítás);
5. Adminisztrátorként való belépéshez: Felhasználó név: admin; Jelszó: admin (kezdeti beállítás);
6. Fejlesztőként lehetősége van a program forráskódjának megtekintésére, az Mx könyvtárban;
7. A program felhasználói és fejlesztői dokumentációja a Dokumentáció könyvtárban található.