|  |
| --- |
| Tutifruty |
| Návrh a špecifikácia komponentov |
| Web občianského združenia FMFI a exkurzie |
|  |
| **Stefan Horvath** |
| **7.11.2012** |

|  |
| --- |
| Tento dokument obsahuje návrh systému a špecifikáciu komponentov. Obhajuje voľbu použítých technológií a návrhových postupov. Definuje vstupné a výstupne parametre metód a špecifikuje ich použitie. Okrem toho obsahuje aj textový a grafický návrh výstupov, ktoré budú zobrazené použivateľskej úrovni. Ďalej odporúča rozdelenie práce pre jednotlivých vývojárov tak, aby bol vývoj čo najefektívnejší. |

# História dokumentu

|  |  |  |  |
| --- | --- | --- | --- |
| Verzia | Autor | Dátum | Vykonané zmeny |
| 0.10 | Štefan Horváth | 7.11.2012 | Prvá verzia |
| 0.20 | Štefan Horváth | 7.12.2012 | Doplnený popis funkcií modelu a závislosti medzi controllermi a views.  Oprava niektorých model funkcií, doplnenie parametrov |
| 0.30 | Matej Toma | 28.01.2013 | - Doplnené dva controllery + popis k týmto controllerom čo vykonávajú.  - Doplnené get\_page(view), get\_login(values) |

# Všeobecné informácie

Účelom toho dokumentu je návrh a špecifikácia komponentov, ktoré budú použité pri zostavovaní release verzie webovej aplikácie Web občianského združenia FMFI a exkurzie (ďalej len „systém“). Je určený pre vývojárov, ktorý budú jednotlivé komponenty implementovať, pričom používajú tento dokument ako referenčný z hladiska špecifikácie. Mali by dodržiavať jednotlivé menné konvencie, ktoré tento dokument definuje. Po implementácii a integrácii týchto komponentov je možné zostaviť kompletnú a spustiteľnú release verziu systému.

Dokument by mal čo najdetailnejšie a najpresnejšie popisovať jednotlivé komponenty, no mohlo sa stať že pri návrhu neboli známe všetky okolnosti alebo sa pozabudlo na nejaký fakt, preto mohli ostať nejaké elementy nešpecifikované, ba dokonca nedefinované. V takomto prípade sa môže vývojár rozhodnúť a požiadať o doplnenie špecifikácie alebo si môže daný element špecifikovať sám.

Dokument obsahuje predľadné tabuľky, podľa ktorých sa vývojári orientujú. Ďalej je popísané rozdelenie práce tak, aby vývoj komponentov bol čo najefektívnejší. K tomuto rozdeleniu práce prislúchajú aj tabulky, ktoré obsahujú štruktúrované informácie tak, aby konkrétny vývojar dostal čo najviac informácii na jednom mieste a nemusel sa prepínať medzi tabuľkami.

K dokumentu patrí príloha, v ktorej je triedny diagram a grafické návrhy jednotlivých stránok.

# Základné rozdelenie komponentov

Na vývoj používame framework nad php – CodeIgniter. Tento framework sme si zvolili aj preto, lebo používa návrhový vzor MVC, teda model – view – controller.

## MVC všeobecne

Tento návrhový vzor nás núti programovať podľa istých pravidiel, ktoré ak dodržime, framework nám značne ulahčí prácu. Je rozdelený na 3 nezávislé komponenty – model, view, controller. Model slúži na komunikáciu z dátovým úložiskom. Obsahuje metódy, ktoré vykonávajú čítanie a zápis dát systému. View obsahuje šablóny, podľa ktorých sa zobrazujú formátované dáta na výstup. Controller slúži ako komunikačná jednotka z používateľom. Podľa požiadavky používateľa controller zobrazí view, pre ktorý pripravý potrebné informácie z modelu. Podľa návrhového vzoru mvc aj model a view vedia spolu komunikovať, view si tak môže sám získať potrebné dáta na zobrazenie z modelu.



MVC definuje aj správu zdrojových súborov a tried. MVC framework obsahuje priečinky Model, View, Controller, kde sa ukladajú zdrojové kódy zodpovedajúce danému názvu. Každý súbor obsahuje práve jednu triedu, ktorej názov je použitý aj v názve súboru. Táto trieda obsahuje všetky metódy a ich implementáciu.

## Základný návrh komponentov a rozdelenie na MVC

Táto sekcia delí špecifické komponenty systému a mapuje ich na návrhový vzor MVC, pričom dodržuje predpísané pravidlá MVC. Z tohto rozdelenia bude vychádzať správa zdrojových súborov, tried a metód.

### Session

Session drží počas práce v systéme údaje o prihlásení používateľa. Tieto údaje musia byť nadradené nad celou MVC štruktúrov, teda ktorýkoľvek komponent musí mať prístup k session. Je to z dôvodu autentifikácie, získania role a osobných údajov o použivateľovi.

### Model

Model slúži na komunikáciu z MySQL databázou. Obsahuje metódy, ktore načítajú údaje z databázy a vrátia ich v podobne požadovanej štruktúry. V prípade jedného zaznamu vráti záznam ako pole, v prípade viacerých záznamov vráti pole objektov. Podrobnejšia špecifikácia jednotlivých funkcií je v ďalších kapitolách.

Tieto metódy sú rozdelené podľa účelu do 4 kontajnérov (tried):

* Selecter
* Inserter
* Updater
* Deleter

Trieda **Selecter** vlastní metódy, ktorých účelom je získavanie dát zo systému.

Trieda **Inserter** vlastní metódy, ktorých účelom je vkladanie dát do systému.

Trieda **Updater** vlastní metódy, ktorých účelom je úprava existujúcich dát v systéme.

Trieda **Deleter** vlastní metódy, ktorých účelom je zmazanie dát zo systému.

Toto delenie bolo navrhnuté kvoli prehľadnosti a kvoli metódam controlleru, z ktorých každá jedna metóda má svoj jedinečný účel a bude používať na prístup k dátam prevažne jeden kontajnér. Pokiaľ by to situácia vyžadovala, môže metóda controlleru použiť aj viac kontajnérov naraz.

Podľa požiadaviek každá vykonaná zmena v databáze bude evidovaná. Pri implementácii treba počitať s tým, že každá metóda kontajnéra okrem svojej definovanej činnosti spraví záznam do logu. Bližšia špecifikácia je v ďalších kapitolách dokumentu.

Každý kontajnér (trieda) bude umiestnený vo vlastnom zdrojovom súbore. Tento súbor bude mať názov „<názov\_kontajnéra>.php“. V tomto súbore bude jedna trieda, ktorá obsahuje všetky metódy patriace k nej.

### View

View je HTML šablóna, ktorá sa zobrazuje na obrazovku. Na dynamické zobrazovanie dát používa premenné, ktorých hodnoty sú inicializované s controllera. K tejto šablóne je pripojený CSS súbor, podľa ktorého webový prehliadač tvorí výsledný dizajn dokumentu.

V systéme bude existovať jeden rodičovský view, ktorý bude volať view predaný parametrom. Tento view predaný parametrom obsahuje samotný obsah. Je to z dôvodu jednotného dizajnu. Rodičovský view obahuje hlavičku a petičku stránky a spoločný dizajn, ktorý je aplikovaný na celú stránku. View predaný parametrom sa zobrazí do rozloženia, ktoré mu určí rodičovský view.

View sú navrhnuté vzhľadom na controlleri. K jednému controlleru existuje niekoľko view. Každá rôzna forma zobrazenia má samostatný view, aj keď sa jedná o tie isté dáta ale inak zobrazené.

Samotný view sa rozhoduje podľa role prihláseného používateľa, aký obsah zobrazí.

Každý view bude definovaný vo vlastnom súbore. Tento súbor bude mať názov: „<názov\_controlleru>\_<názov\_metódy>\_view.php“. V súbore bude celá šablóna, ktorá formátuje výstup z premenných. Tieto premenné sú definované v dokumente nižšie.

#### Grid

Často budeme zobrazovať dáta v forme tabuľky. Aby nemusel každý view definovať svoju vlastnú tabuľku na zobrazenie dát, použijeme nezávislí komponent *grid*. Pomôžeme tým dosiahnuť jednotný dizajn, zlepšíme efektivitu vývoja a eliminujeme množstvo chýp, ktoré by môhli vzniknúť.

Tento *grid* je definovaný ako trieda. View, ktorý tento grid používa vytvára jeho inštanciu, naplní ho požadovanými dátami a násedne zavolá metódu na zobrazenie. Komponent je narhnutý univerzálne tak, aby bolo možné jeho všestranné použitie či už na zobrazenie, alebo na kompletný CRUD. Mal by obsahovať rôzne možnosti prispôsobenia, či už dátove alebo vizuálne.

#### Confirm

Často bude využívaná otázka, ktorá má zistiť či je používateľ naozaj rozhodnutý vykonať danú operáciu a je si vedomí všetkých jej následkov. Na tento účel bude vytvorená funkcia, ktorá zobrazí naformátovanú otázku a ponúkne odpovede typu „áno“, „nie“. Táto otázka bude nezávislá a všeobecne použiteľná. Výhody sú podobné ako pri komponente *grid*.

### Controller

Controller slúži ako komunikačný prostriedok medzi použivateľom a systémom. Z programátorského hladiska je controller trieda, ktorá obsahuje metódy s parametrami. Používateľ volá metódy daného controllera pomocou url adresy webového prehliadača. Štandartne sa mapuje url adresa na controller ako: *...doména.sk/<controller>/<metóda>/<parameter1>/<parameter2>/...*

Controller zhromaždí údaje z databázy cez kontajnéry modelu, inicializuje premenné view a následne zobrazí rodičovský view ktorému pošle ako parameter požadovaný obsahový view. Inicializácia premenných obsahového view prebieha cez asociatívne pole $data. Toto pole sa skladá z indexu a hodnoty, pričom ako index sa používa string. Indexy sa vo view mapujú ako názvy premenných a hodnoty sú ako hodnoty premenných.

príklad:

Controller:

$data[‘**meno’**] = ‘Jozo’

$data[‘**vek’**] = 22

View:

echo **$meno**;

echo **$vek**;

Prvok v poli *data* nemusí byť len jednoduchá hodnota, môže to byť aj zložitejší typ ako napr. pole. V systéme budeme často do view predávať zoznamy, ktoré budú reprezentované ako pole objektov, pričom objekt može byť tiež ako pole.

Metóda controlleru musí na začiatku vykonávania skontrolovať práva používateľa a podľa toho rozhodnúť, či bude pokračovať vo vykonávaní alebo ukončí vykonávanie.

Controllery delíme tématicky do následujúcich kategórií:

* administration
* auth
* correspondence
* degrees
* email\_types
* event\_categories
* events
* excursion\_events
* excursions
* io
* pages
* payments
* posts
* project\_categories
* projects
* studies
* users

Trieda **administration** slúží na obsluhovanie plne administratývnych záležitostí a prístupu k funkciam administrácie.

Trieda **auth** slúži na registráciu a správu autorizácie použivateľov.

Trieda **correspondence** slúži na hromadné odosielanie emailov.

Trieda **degrees** slúži na správu dosiahnutých titulov.

Trieda **email\_types** slúži na správu typov e-mailov.

Trieda **event\_categories** slúži na správu kategórií udalostí.

Trieda **events** slúži na správu udalostí, vie zobraziť udalosti podľa určitéh filtra.

Trieda **excursion\_events** slúži na správu udalosí exkurzií, umožnuje prihlasovanie sa na udalosť, definovanie prednášajúcich a ich termínov.

Trieda **excursions** slúži na správu exkurzií.

Trieda **IO** slúži na generovanie reportov a výstupov podľa zvolených kritérií a import údajov do systému.

Trieda **pages** slúží na obsluhovanie statických stránok akými sú stanovy, o nás, kontakt.

Trieda **payments** slúži na správu platieb, umožnuje párovanie platieb k členom.

Trieda **posts** slúži na správu článkov.

Trieda **project\_categories** slúži na správu kategórií projektov, manžuje financie, umožnuje finančné transakcie medzi kategóriami.

Trieda **projects** slúži na správu projektov.

Trieda **studies** slúži na spŕavu študijných programov.

Trieda **users** slúži na správu používateľov.

\*Tieto kategórie definujú tiež konkrétne názvy tried controllerov.

# Podrobnejšia špecifikácia komponentov

## Session

Systém používa session integrovaný priamo v CodeIgniter frameworku. Pri prihlásení používatela je nutné aktivovať tento session a inicializovať ho potrebnými hodnotami pomocou pravidiel CodeIgniteru. Rovnako ho treba zrušiť pri odhlásení.

Získanie dát o práve prihlásenom používateľovi sa robí nasledovne:

$this->userdata-><dotaz>();

pričom *dotaz* je nahradený podľa toho, aký udaj chceme získať.

### Dotazy

* is\_logged : bool
* is\_admin : bool
* get\_role : int {ROLE\_ADMIN, ROLE\_OZ\_MEMBER, ROLE\_EX\_MEMBER, ROLE\_LECTURER}

## Model

Vysvetlenie návratových hodnôt metód:

* object - metóda vracia jeden riadok tabuľky
* array of object - metóda vracia viac riadkov tabuľky, resp. pole objektov

Priklad object:

*$result = get\_user(1);*

*echo $result[‘name’];*

*echo $result[‘age’]*

Priklad array of object:

*$result = get\_users()*

*echo $result[0][‘name’];*

*echo $result[0][‘age’];*

*echo $result[1][‘name’];*

*echo $result[1][‘age’];*

*\*ideálne prechádzať result pomocou* ***foreach***

Pri návratových hodnotách sú v tabuľke definované len indexy objektov. Polia objektov sú vždy od 0 po *počet\_riadkov*.

Vysvetlenie vstupných parametrov metód:

* object - metóda prijíma jeden riadok definovaný ako pole
* array of objects - metóda prijíma viac riadkov, resp. viac objektov definovaných ako pole objektov
* array - metóda prijíma elementárne prvky (int, string, ...) ako pole

V nasledujúcich tabuľkách je zoznam databázových funkcií, ktoré bude používať systém na správu dát. Prvý stĺpec obsahuje presný názov funkcie. Druhý stĺpec definuje vstupné alebo výstupné premenné podľa typu funkcie. Tretí stĺpec v prípade netriviálnej situácie upresňuje správanie funkcie, prípadne hovorí o postupe získania požadovaných dát. Každá funkcia má vo svojom pomenovaní aj názov tabuľky, z ktorej dáta vyberá/ vkladá.

Indexy polí alebo atribúty objektov, ktoré funkcie selecter-u vracajú, sú väčšinou zhodné z názvami stĺpcov tabuliek v databáze. Ak by sa nejaký názov nezhodoval, znamená to že je iný z nejakého dôvodu a tým pádom treba tento stĺpec v SQL dopyte premenovať. Funkcie, ktoré prijímajú nejaké parametre majú zasa indexy/ atribúty prevažne zhodné z formulármi z views, z ktorých tieto vstupné dáta pochádzajú. Návrh sa v čo najväčšej možnej miere snaží zachovať rovnaké pomenovania aj pre stĺpce tabuliek v databáze, aj pre formulárové elementy. Tým by sa mal minimalizovať proces premenovávania.

Ak tabuľka databázy obsahuje viac stĺpcov ako príde vstupných parametrov (indexov, atribútov), zapíše sa do nedefinovaných stĺpcov tabuľky informáciu typu *null*.

### Selecter

Nasledujúce funkcie majú za úlohu vrátiť nejaké konkrétne údaje z databázy. Väčšinou vyberajú údaje z jednej tabuľky, pričom k týmto záznamom dohľadávajú údaje z iných tabuliek podľa id-čka. Ak je funkcia netriviálna (teda vyberá údaje z viac ako jednej tabuľky), jej bližšia definícia sa nachádza v stĺpci „popis“. V prípade triviálnych funkcií sa urobí jednoduchý výber z databázy. Názov funkcie (bez slovíčka „get“) hovorí aj o názve tabuľky, z ktorej sa údaje vyberajú.

Funkcie vracajú objekt (alebo pole objektov). Objekt obsahuje atribúty pomenované tak, ako sú definované v stĺpci „výstupy“. Vo väčšine prípadov toto pomenovanie vytvorí priamo výstup z databázy, nakoľko veľa krát je pomenovanie zhodné z pomenovaním stĺpcov tabuliek.

|  |  |  |
| --- | --- | --- |
| Názov funkcie | Výstupy | Popis |
| get\_category\_detail(cat\_id) | return : object ->  [project\_category\_id] : int  [project\_category\_name] : string  [project\_category\_cash] : float  [transaction\_cash\_from] : float  [transaction\_cash\_to] : float | výber konkrétneho záznamu z tabuľky *project\_categories*, na základne id-čka vytiahnuť z tabuľky *fin\_category\_transaction* sumár presunov from a to |
| get\_degrees(grid = false) | return : array of objects  [degree\_id] : int  [degree\_name] : string  [degree\_grade] : int | Vráti všetky záznamy z tabuľky degrees a zobrazí ich do zobrazovacieho gridu. Ak je vstupný parameter *true* tak v prípade žiadnych záznamov sa v gride objavia hlavičky stĺpcov aby bolo možné pridať nové záznamy. Defaultne je nastaveny na *false*. |
| get\_email\_types(grid = false) | return : array of objects  [email\_type\_id] : int  [email\_type\_name] : string | Vráti všetky záznamy z tabuľky email\_types a zobrazí ich do zobrazovacieho gridu. Ak je vstupný parameter *true* tak v prípade žiadnych záznamov sa v gride objavia hlavičky stĺpcov aby bolo možné pridať nové záznamy. Defaultne je nastaveny na *false*. |
| get\_event\_categories(grid = false) | return : array of objects ->  [event\_category\_id] : int  [event\_category\_name] : string | Vráti všetky záznamy z tabuľky event\_categories a zobrazí ich do zobrazovacieho gridu. Ak je vstupný parameter *true* tak v prípade žiadnych záznamov sa v gride objavia hlavičky stĺpcov aby bolo možné pridať nové záznamy. Defaultne je nastaveny na *false*. |
| get\_event\_detail(event\_id) | return : object ->  [event\_category\_id] : int  [event\_category] : string  [event\_name] : string  [event\_about] : string  [event\_from] : datetime  [event\_to] : datetime  [event\_priority] : int  [event\_author\_id] : int  [event\_author] : string  [event\_created] : datetime | Vráti záznam z tabuľky events pričom k tomuto záznamu získa z tabuľky users meno používateľa, ktorý daný event vytvoril a to na základe *event\_author\_id,* ktorý sa nachádza v tabuľke events. |
| get\_events(per\_page=0, cur\_page=0, cat\_id, grid=false) | return : array of objects ->  [event\_id] : int  [event\_about] : string  [event\_name] : string  [event\_created] : datetime  [event\_category\_id] : int  [event\_category\_name] : string  [event\_priority] : int  [event\_from] : datetime  [event\_to] : datetime | výber záznamov z tabuľky *events*, na základne id-čka vytiahnuť z tabuľky *event\_categories* názov a id-čko kategórie, do ktorej daný event patrí |
| get\_events\_newest(per\_page=0, cur\_page=0, cat\_id, grid=false) | return : array of objects ->  [event\_id] : int  [event\_about] : string  [event\_name] : string  [event\_created] : datetime  [event\_category\_id] : int  [event\_category\_name] : string  [event\_priority] : int  [event\_from] : datetime  [event\_to] : datetime | výber záznamov z tabuľky *events*, na základne id-čka vytiahnuť z tabuľky *event\_categories* názov a id-čko kategórie, do ktorej daný event patrí  vracia záznamy z tabuľky *events* zoradené podľa stĺpca *event\_from* od najskoršieho dátumu. |
| get\_events\_prior(per\_page=0, cur\_page=0, cat\_id, grid=false) | return : array of objects ->  [event\_id] : int  [event\_about] : string  [event\_name] : string  [event\_created] : datetime  [event\_category\_id] : int  [event\_category\_name] : string  [event\_priority] : int  [event\_from] : datetime  [event\_to] : datetime | výber záznamov z tabuľky *events*, na základne id-čka vytiahnuť z tabuľky *event\_categories* názov a id-čko kategórie, do ktorej daný event patrí  vracia záznamy z tabuľky *events* zoradené podľa stĺpca *event\_priority* od najväčšej priority. |
| get\_excursion\_detail(excursion\_id) | return : object ->  [excursion\_name] : string  [excursion\_about] : string  [excursion\_priority] : int  [excursion\_from] : datetime  [excursion\_to] : datetime |  |
| get\_excursion\_event\_book(ex\_ev\_book\_id) | return : object ->  [excursion\_name] : string  [excursion\_event\_excursion] : string  [booked\_excursion\_number\_of\_visitors] : int | výber konkrétneho záznamu z tabuľky *booked\_excursion*, dohladať z tabuľky *excursion\_events* názov eventu a z *excursion* nazov exkurzie |
| get\_excursion\_event\_detail(ex\_event\_id) | return : object ->  [excursion\_id] : int  [excursion\_name] : string  [excursion\_event\_name] : string  [excursion\_event\_place] : string  [excursion\_event\_room] : string  [excursion\_event\_abstract] : string  [excursion\_event\_about] : string  [excursion\_event\_special\_requirements] : string  [excursion\_event\_active] : int  [excursion\_event\_max\_people] : int | výber konkrétneho záznamu z tabuľky *excursion\_events*, dohladať z tabuľky *excursion* názov exkurzie a id-čko exkurzie |
| get\_excursion\_event\_free(excursion\_event\_id) | return : int | z tabuľky *booked\_excursion* urobiť na základe zadaného id-čka sumár návštevníkov, z tabuľky *excursion\_events* získať kapacitu, na základe týchto informácie vrátiť počet voľných miest |
| get\_excursion\_event\_lecturers(ex\_event\_id) | return : array of object ->  [user\_id] : int  [user\_name] : string  [user\_surname] : string  [degree\_name] : string | výber záznamov z tabuľky *users*, dohladať z tabuľky *degrees* názov titulu |
| get\_excursion\_event\_visitors(ex\_event\_id) | return : array of object ->  [user\_id] : int  [user\_name] : string  [user\_surname] : string  [user\_email] : string  [booked\_excursion\_number\_of\_visitors] : int | výber záznamov z tabuľky *users*, dohladať z tabuľky *booked\_excursion* názov počet návštevníkov |
| get\_excursion\_events(excursion\_id) | return : array of object ->  [excursion\_event\_excursion\_id] : int  [excursion\_event\_excursion] : string  [excursion\_event\_id] : int  [excursion\_event\_name] : string  [excursion\_event\_place] : string  [excursion\_event\_room] : string  [excursion\_event\_from] : string  [excursion\_event\_to] : string | výber záznamov z tabuľky *excursion\_events*, dohladať z tabuľky *excursion* názov a id-čko exkurzie |
| get\_excursions() | return : array of object ->  [excursion\_id] : int  [excursion\_name] : string  [excursion\_from] : date  [excursion\_to] : date |  |
| get\_excursions(from, to) | return : array of object ->  [excursion\_id] : int  [excursion\_name] : string  [excursion\_from] : date  [excursion\_to] : date | viď get\_excursions(), roziel je že táto funkcia vracia exkurzie v definovanom časovom rozsahu |
| get\_excursions\_for\_lecturer(user\_id) | return : array of object ->  [excursion\_name] : string  [excursion\_event\_id] : int  [excursion\_event\_name] : string  [excursion\_time\_from] : datetime  [excursion\_time\_to] : datetime | výber usera z tabuľky *users*, k tomuto záznamu vybrať z tabuľky *excursion\_events* názvy a id-čká eventov exkurzií, k týmto záznamom dohladať z tabuľky *excursion\_times* časy |
| get\_excursions\_for\_visitor(user\_id) | return : array of object ->  [excursion\_name] : string  [excursion\_event\_id] : int  [excursion\_event\_name] : string  [booked\_excursion\_number\_of\_visitors] : int | výber usera z tabuľky *users*, k tomuto záznamu vybrať z tabuľky *excursion\_events* názvy a id-čká eventov exkurzií, k týmto záznamom dohladať z tabuľky *booked\_excursions* počet návštevníkov |
| get\_fin\_redistribution(payment\_id, pr\_cat\_id) | return : array of objects ->  [fin\_redistribute\_id] : int  [fin\_redistribute\_ratio] : int | záznam vyhovuje, aj sú obe id-čka z tabuľky zhodné s parametrami |
| get\_last\_clean() |  | MY\_controller |
| get\_login(values) | return: object ->  [user\_id] : int  [user\_role] : int | Vráti jeden záznam z tabuľky *users.* Jedná sa o prihláseného človeka. |
| get\_lecturer\_times(ex\_event\_id, user\_id) | return : array of object ->  [excursion\_time\_from] : datetime  [excursion\_time\_to] : datetime | záznam vyhovuje, aj sú obe id-čka z tabuľky zhodné s parametrami |
| get\_lecturers() | return : array of object ->  [user\_id] : int  [user\_name] : string |  |
| get\_page(view) | return: object->  [page\_view] : text | Vráti jeden stĺpec z tabuľky *pages*, ktorá uchováva informácie o stanovách, o nás, kontakt. View je názov stránky o ktorej chceme zýskať údaje. |
| get\_payment\_detail(payment\_id) | return : object ->  [payment\_id] : int  [payment\_vs] : int  [payment\_total\_sum] : float  [payment\_paid\_sum] : float  [payment\_user\_id] : int  [payment\_paid\_time] : datetime  [payment\_type] : int  [categories][cat\_id] : array of int | vytiahne z tabuľky *payments* platbu podľa *payment\_id*, k nej všetky záznamy z tabuľky *fin\_redistributes*. |
| get\_payments(per\_page=0, cur\_page=0, user\_id, grid = false) | return : array of objects ->  [user\_id] : int  [user\_name] : string  [user\_surname] : string  [payment\_type]: int  [payment\_vs] : int  [payment\_total\_sum] : float  [payment\_paid\_sum] : float  [payment\_paid\_time] : datetime  [payment\_accepted] : tinyint | vytiahne usera z *users* a k nemu platby z *payments*, ak je user\_id == 0, vytiahne všetkych userov a k nim platby. Parameter *per\_page*  udáva koľko záznamov má byť zobrazených na jednu stránku. Parameter *cur\_page* hovorí o tom aká je aktuálna strana. *Grid* s parametrom false vráti výsledok ako array of objects. Ak je true tak výsledok je v podone CI\_DB\_Object. |
| get\_payments\_lastpaid(user\_id) | return : object ->  [payment\_paid\_time] : datetime  [payment\_type]: int  [payment\_paid\_sum] : float  [payment\_total\_sum] : float | vyberie z tabuľky *payments* poslednú uradenú platbu usera podľa user\_id |
| get\_payments\_nopaid(per\_page=0, cur\_page=0, user\_id, grid = false) | return : array of objects ->  [user\_id] : int  [user\_name] : string  [user\_surname] : string  [payment\_type]: int  [payment\_vs] : int  [payment\_total\_sum] : float  [payment\_paid\_sum] : float  [payment\_paid\_time] : datetime  [payment\_accepted] : tinyint | Ak bude vstupný parameter user\_id rovný 0, vracia **všetky** neuhradené platby, ak bude iné číslo, vracia všetky neuhradené platby **konkrétneho usera.**  Údaje sa vyberajú z tabuľky *payments*, podľa payment\_user\_id sa dohľadá z tabuľky *users* meno a priezvisko usera. Parameter *per\_page*  udáva koľko záznamov má byť zobrazených na jednu stránku. Parameter *cur\_page* hovorí o tom aká je aktuálna strana. *Grid* s parametrom false vráti výsledok ako array of objects. Ak je true tak výsledok je v podone CI\_DB\_Object. |
| get\_payments\_paid(per\_page=0, cur\_page=0, user\_id, grid = false) | return : array of objects ->  [user\_id] : int  [user\_name] : string  [user\_surname] : string  [payment\_type]: int  [payment\_vs] : int  [payment\_total\_sum] : float  [payment\_paid\_sum] : float  [payment\_paid\_time] : datetime  [payment\_accepted] : tinyint | Ak bude vstupný parameter user\_id rovný 0, vracia **všetky** uhradené platby, ak bude iné číslo, vracia všetky uhradené platby **konkrétneho usera.**  Údaje sa vyberajú z tabuľky *payments*, podľa payment\_user\_id sa dohľadá z tabuľky *users* meno a priezvisko usera. Parameter *per\_page*  udáva koľko záznamov má byť zobrazených na jednu stránku. Parameter *cur\_page* hovorí o tom aká je aktuálna strana. *Grid* s parametrom false vráti výsledok ako array of objects. Ak je true tak výsledok je v podone CI\_DB\_Object. |
| get\_post\_detail(post\_id) | return : object ->  [post\_title] : string  [post\_content] : string  [post\_published] : tinyint  [post\_author\_id] : int  [post\_date] : datetime  [post\_modifie\_author\_id] : int  [post\_modifie\_date] : datetime  [author\_name] : string  [author\_surname] : string  [modifie\_name] : string  [modifie\_surname] :string | Z tabuľky *posts*  vráti jeden konkrétny záznam (príspevok). K týmto údajom sa naviaže z tabuľky *users* informácie o tom kto daný príspevok vytvoril [*post\_author\_id*] jeho meno a priezvisko zo spomínanej tabuľky a výsledný stĺpec ktorý takéto meno obsahuje je [*author\_name]*. A taktiež ak existuje k danému príspevku aj nejaký jeho modifikátory, ktorí daní príspevok upravovali. Tak tieto záznamy sa vytiahnu z tabuľky *post\_modifies* a na základe *post\_modifie\_post\_id* sa získa z tabuľky *users* meno a priezvisko upravovateľa. |
| get\_post\_modifiers(post\_id) | return : array of objects ->  [user\_id] : int  [user\_name] : string  [user\_surname] : string  [post\_modifie\_id] : int  [post\_modifie\_date] : datetime | Z tabuľky *post\_modifies* vybrať všetky záznamy na základe zadaného post\_id, k týmto záznamom doplniť podľa post\_modifie\_author\_id údaje z tabuľky *user*. |
| get\_posts(per\_page=0, cur\_page=0, unpublished=false) | return : array of objects ->  [post\_id] : int  [post\_title] : string  [post\_content] : string  [post\_author\_id] : int  [post\_priority] : int  [post\_date] : datetime  [post\_modifie\_author\_id] : int  [post\_modifie\_date] : datetime  [modifier\_name] : string  [modifier\_surname] : string  [post\_published] : tinyint  [author\_name] : string  [author\_surname] : string | Z tabuľky posts treba vybrať všetky príspevky a k ním všetky údaje, ktoré sa k takémuto príspevku viažu. Takto vytiahnuté príspevky treba usporiadať zostupne podľa *post\_date*. K týmto údajom sa naviaže z tabuľky *users* informácie o tom kto daný príspevok vytvoril [*post\_author\_id*] jeho meno a priezvisko zo spomínanej tabuľky a výsledný stĺpec ktorý takéto meno obsahuje je [*author\_name]*. A taktiež ak existuje k danému príspevku aj nejaký jeho modifikátory, ktorí daní príspevok upravovali. Tak tieto záznamy sa vytiahnu z tabuľky *post\_modifies* a na základe *post\_modifie\_post\_id* sa získa z tabuľky *users* meno a priezvisko upravovateľa. Parameter *per\_page*  udáva koľko záznamov má byť zobrazených na jednu stránku. Parameter *cur\_page* hovorí o tom aká je aktuálna strana. *Grid* s parametrom false vráti výsledok ako array of objects. Ak je true tak výsledok je v podone CI\_DB\_Object. Parameter *unpublished* zabezpečí to, že či chceme iba publikované alebo všetky príspevky. |
| get\_project\_categories(grid=false) | return : array of objects [project\_category\_id] : int  [project\_category\_name] : string  [project\_category\_cash] : float  [booked\_cash] : float | Vráti všetky záznamy z tabuľky project\_categories a zobrazí ich do zobrazovacieho gridu. Ak je vstupný parameter *true* tak v prípade žiadnych záznamov sa v gride objavia hlavičky stĺpcov aby bolo možné pridať nové záznamy. Defaultne je nastaveny na *false*. |
| get\_project\_categories\_total\_cash() | return -> [total\_sum] : float | Vráti sumár všetkých záznamov z tabuľky *project\_categories* podľa stĺpca project\_category\_cash |
| get\_project\_detail(project\_id) | return : object ->  [project\_name] : string  [project\_about] : string  [project\_priority] : int  [project\_project\_category\_id] : int  [project\_booked\_cash] : float  [project\_spended\_cash] : float  [project\_category\_name] : string  [project\_date\_from] : date  [project\_date\_to] : date  [project\_active] : tinyint | Vráti všetky informácie ktoré sa viažu k projektu s daným project\_id. Základnou tabuľkou je project\_items čím získame informácie o daných itemoch ktoré patria pod daný project. K nim naviažeme tabuľku projects z ktorej získame všetky požadované informácie, ktoré sa viažu ku projektu. |
| get\_project\_items(project\_id, grid = false) | return : array of objects  [project\_item\_name] : string  [project\_item\_price] : float  [user\_id] : int  [user\_name] : string  [user\_surname] : string  [project\_item\_id] : int  [project\_item\_date] : date | Vráti záznamy z tabuľky *project\_items* podľa parametra project\_id, z tabuľky *users* doplní meno k týmto záznamov id, meno a priezvisko usera (podľa project\_item\_user\_id) |
| get\_projects(cat\_id, grid = false) | return : array of objects ->  [project\_id] : int  [project\_name] : string  [project\_booked\_cash] : float  [project\_spended\_cash] : float  [project\_date\_from] : date  [project\_date\_to] : date  [project\_category\_id] : int  [project\_category\_name] : string  [project\_item\_id] : int  [project\_active] : tinyint | Vráti všetky záznamy z tabuľky projects na základe parametra *cat\_i.*  Ak je parameter *cat\_id* = 0 potom treba vrátiť všetky záznamy z tabuľky projects v inom prípade tie ktoré maju *project\_project\_category\_id* také aký je vstupný parameter. Z danej tabuľky vybere ku záznamom aj potrebné stĺpce tak ako je to špecifikované. *Grid* s parametrom false vráti výsledok ako array of objects. Ak je true tak výsledok je v podone CI\_DB\_Object. |
| get\_study\_programs(grid = false) | return : array of objects ->  [study\_program\_id] : int  [study\_program\_name] : string | Vráti všetky záznamy z tabuľky study\_programs a zobrazí ich do zobrazovacieho gridu. Ak je vstupný parameter *true* tak v prípade žiadnych záznamov sa v gride objavia hlavičky stĺpcov aby bolo možné pridať nové záznamy. Defaultne je nastaveny na *false*. |
| get\_transactions(pr\_cat\_id) | return : array of objects ->  [fin\_category\_transaction\_id] : int  [fin\_category\_transaction\_cat\_from] : int  [fin\_category\_transaction\_cat\_to] : int  [fin\_category\_transaction\_cat\_cash] : float  [fin\_category\_transaction\_cat\_to\_id] : int  [fin\_category\_transaction\_cat\_from\_id] : int | Vráti všetky záznamy z tabuľky fin\_category\_transactions. Tieto záznamy slúžia na informáciu z kade kam boli prevedené aké množstvo financií vrámci projektových kategórií. Keďže tabuľka obsahuje ID kategórie z kade boli odčerpané peniaze a ID kategórie kam boli presunuté peniaze tak treba spraviť dve samostatné sub-query kde sa k týmto ID-čkam získa meno danej kategórie. Spraviť agregačnú funkciu súčtu a výsledky oboch sub-queries spojiť do výsledného resultu. |
| get\_user\_detail(user\_id) | return : object ->  [user\_id] : int  [user\_date] : date  [user\_postcode] : int  [user\_study\_program\_id] : int  [user\_study\_program] : string  [user\_degree\_id] : int  [user\_degree] : string  [user\_place\_of\_birth] : string  [user\_degree\_year] : int  [user\_role] : int  [user\_username] : string  [user\_name] : string  [user\_surname] : string  [user\_password] : string  [user\_email] : string  [user\_phone] : int  [user\_activated] : int | Vráti záznam z tabuľky users na základe *user\_id* čo je ID používateľa o ktorom chceme získať informácie z databázy. |
| get\_users(per\_page = 0, cur\_page = 0, role, grid = false) | return : array of objects ->  [user\_id] : int  [user\_name] : string  [user\_email] : string  [user\_phone] : int  [user\_degree\_year] : int  [study\_program\_name] : string  [degree\_name] : string  [user\_postcode] : int | Ak je parameter role rovný 0, vráti **všetkých** userov, ak je rôzny od 0, vyberie userov ktorí vyhovujú zadanej roli.  Záznamy sa vyberajú z tabuľky *users*, ku každému záznamu sa dohľadá študijný program z tabuľky study\_*programs* a stupeň vzdelania z tabuľky *degrees*. Parameter *per\_page*  udáva koľko záznamov má byť zobrazených na jednu stránku. Parameter *cur\_page* hovorí o tom aká je aktuálna strana. *Grid* s parametrom false vráti výsledok ako array of objects. Ak je true tak výsledok je v podone CI\_DB\_Object. |
| Get\_users\_filter(values) | return : array of objects ->  [user\_name] : string  [user\_surname] : string  [user\_degree] : string  [user\_study\_program] : string  [user\_email\_evidence\_date] : date  [email\_type\_name] : string  [user\_email] : string  input values: array ->  [study][0..n-1] : int  [degree][0..m-1] : int  [degree\_year][0..p-1] : int  *\*n – počet id-čiek študijných programov*  *\*m – počet stupnov vzdelania*  *\*p – počet rokov ukončenia štúdia* | Vstupné parametre je pole polí integerov. **Study** sú indexy študijných programov, **degree** sú stupne vzdelania, **degree\_year** sú roky ukončenia štúdia. Tieto id-čka boli zvolené userom na stránke.  Vyberajú sa záznamy z tabuľky *users*. Zo vstupných parametrov je nutne vygenerovať časť SQL za klauzulou WHERE tým, že sa prejdú vstupné parametre cez *foreach* a ich hodnoty sa dajú do porovnania s príslušnými stĺpcami tabuliek.  Hodnoty **study\_ids** sa porovnávajú s *users.user\_study\_program\_id*.  Hodnoty **degree\_years** sa porovnávajú s *users.user\_degree\_year*.  Hodnoty z **degrees** sa použijú na získanie záznamov z tabuľky degrees podľa stĺpca *degree\_grade*. Zo získaných údajov sa získané id-čka použijú na porovnanie s *users.user\_degree\_id* podobne ako vyššie.  Ku každému záznamu treba z tabuľky *user\_email\_evidence* vytiahnuť dátum, kedy bol naposledy poslaný email a k tomuto emailu treba ešte z tabuľky *email\_types* vytiahnuť názov.  V princípe funkcia vráti userov podľa zvoleného filtra od používateľa. |

### Inserter

Nasledujúce funkcie majú za úlohu vytvárať nové záznamy v databáze. Štandardne funkcie prijímajú ako parameter „values“ pole hodnôt získaných z formulára na html stránke. Tieto hodnoty musia byť v poli uložené pod indexami, ktoré sú definované v stĺpci „vstup“. Aby sa dosiahlo toto pomenovanie, je doporučené pomenovať takto elementy formulára na html stránke.

|  |  |  |
| --- | --- | --- |
| Názov funkcie | Vstup | Popis |
| add\_degree(values) | values : object ->  [degree\_name] : string  [degree\_grade] : int |  |
| add\_email\_log(email\_type\_id, user\_ids) | email\_type\_id : int  user\_ids : array of int | user\_ids je array id-ciek userov, ktorým bol poslaný hromadný email, treba urobiť záznam do tabuľky *user\_email\_evidence* pre každý user\_id |
| add\_email\_type(values) | values : object ->  [email\_type\_name] : string |  |
| add\_event(values) | values : object ->  [event\_category\_id] : int  [priority] : int  [name] : string  [from] : datetime  [to] : datetime  [about] : string | stĺpec event\_author\_id získame zo SESSION |
| add\_event\_category(values) | values : object ->  [event\_category\_name] : string |  |
| add\_excursion(values) | values : object ->  [name] : string  [about] : string  [priority] : int  [from] : datetime  [to] : datetime |  |
| add\_excursion\_event(values) | values : object ->  [name] : string  [about] : string  [abstract] : string  [lecturer\_id] : int  [from] : datetime  [to] : datetime  [room] : string  [place] : string  [special\_requirements] : string  [active] : int |  |
| add\_excursion\_event\_book(values) | values : object ->  [ex\_event\_id] : int  [visitors] : int | booked\_excursion\_user\_id získame zo SESSION |
| add\_excursion\_event\_lecturer(ex\_event\_id, values) | ex\_event\_id : int  values : object ->  [user\_id] : int | vkladá záznam do tabuľky *excursion\_times*, **user\_id** -> *excursion\_time\_excursion\_id*, **ex\_event\_id** -> *excursion\_time\_excursion\_event\_id*, ostatné stĺpce tabuľky sú **null**. |
| add\_excursion\_event\_visitor(values) | values : object ->  [ex\_event\_id] : int  [user\_id] : int  [visitors] : int | vkladá záznam do tabuľky *booked\_excursions* |
| add\_lecturer\_times(ex\_event\_id, user\_id, values) | ex\_event\_id : int  user\_id : int  values : object ->  [from] : datetime  [to] : datetime | viď *add\_excursion\_event\_lecturer(ex\_event\_id, values)*, táto funkcia ale obsahuje všetky potrebné parametre, teda žiaden stĺpec nebude **null**. |
| add\_payments(values) | values : object ->  [user\_id] : int  [payment\_type] : int  [total\_sum] : float  [payment\_vs] : float  [categories][cat\_id] : int | vkladá záznam do tabuľky *payments*, sum sa uloží na stĺpec total\_sum. *Paid\_sum* bude 0. *Categories* je asoc. Pole, ktoré treba prejsť cez *foreach*. V každej iterácií treba údaj z [categories][<prechadzany\_index>] použiť na nový záznam do tabuľky fin\_redistribution , kde prechádzaný index ide do stĺpca *project\_category\_id* a údaj na ňom ide do stĺpca *ratio*. *Payment\_id* je id-čko práve pridanej platby. |
| add\_post(values) | values : object ->  [title] : string  [content] : string  [priority] : int  [post\_published] : tinyint | okrem vstupných parametrov sa do tabuľky *posts* vypĺňa aj stĺpec post\_author\_id zo SESSION a post\_date aktuálnym dátumom, ostatné sú **null**. |
| add\_project(values) | values : object ->  [name] : string  [about] : string  [priority] : int  [project\_category\_ id] : int  [booked\_cash] : float  [from] : date  [to] : date |  |
| add\_project\_category(values) | values : object ->  [project\_category\_name] : string |  |
| add\_project\_item(pr\_id, values) | pr\_id : int  values : object ->  [name] : string  [price] : float  [user\_id] : int  [Y-m-d] : date |  |
| add\_register(values) | values : object ->  [username] : string  [name] : string  [surname] : string  [password] : string  [email] : string  [phone] : string  [study\_program\_id] : int  [degree\_id] : int  [place\_of\_birth] : string  [postcode] : int  [degree\_year] : int  [categories][cat\_id] : int | categories je pole dĺžky <počet\_kategórií\_projektov>. Funkcia vytvorí záznam v tabuľke *users*, okrem toho preleze pole *categories* pomocou foreach a zapíše hodnoty do tabuľky *fin\_redistribute*, pričom *kľúč* je sa zapíše do stĺpca *fin\_redistribute\_project\_category\_id*, *hodnota* sa zapíše do stĺpca *fin\_redistribude\_ratio*. |
| add\_study\_program(values) | values : object ->  [study\_program\_name] : string |  |
| add\_transaction(values) | values : object ->  [from] : int  [to] : int  [cash] : float | záznam do tabuľky *fin\_category\_transaction* |
| add\_user(values) | values : object ->  [username] : string  [name] : string  [surname] : string  [password] : string  [email] : string  [phone] : int  [study\_program\_id] : int  [degree\_id] : int  [place\_of\_birth] : string  [postcode] : int  [degree\_year] : int  [vs] : int  [total\_sum] : int  [categories][cat\_id] : int | viď add\_register... jediný rozdiel je, že pole categories môže byť nulovej dĺžky |

### Updater

Nasledujúce metódy majú podobný význam ako metódy v insertery, len nevytvárajú nový záznam ale upravujú existujúci. Existujúci záznam sa štandardne hľadá podľa prvého parametra funkcie (id-čka). Pre bližšiu špecifikáciu viď ekvivalentnú funkciu v inserteri. V prípade odlišnosti od insertera je táto zmena uvedená v popise.

Štandardne funkcie prijímajú ako parameter „values“ pole hodnôt získaných z formulára na html stránke. Tieto hodnoty musia byť v poli uložené pod indexami, ktoré sú definované v stĺpci „vstup“. Aby sa dosiahlo toto pomenovanie, je doporučené pomenovať takto elementy formulára na html stránke.

|  |  |  |
| --- | --- | --- |
| Názov funkcie | Vstupy | Popis |
| edit\_degree(degree\_id, values) | degree\_id : int  values : object ->  [degree\_name] : string  [degree\_grade] : int |  |
| edit\_email\_type(e\_type\_id, values) | e\_type\_id : int  values : object ->  [email\_type\_name] : string |  |
| edit\_event(ev\_id, values) | ev\_id : int  values : object ->  [event\_category\_id] : int  [priority] : int  [name] : string  [from] : datetime  [to] : datetime  [about] : string |  |
| edit\_event\_category(ev\_cat\_id, values) | ev\_cat\_id : int  values : object ->  [event\_category\_name] : string |  |
| edit\_excursion(ex\_id, values) | ex\_id : int  values : object ->  [name] : string  [about] : string  [priority] : int  [from] : datetime  [to] : datetime |  |
| edit\_excursion\_event(ex\_ev\_id, values) | ex\_ev\_id : int  values : object ->  [name] : string  [about] : string  [abstract] : string  [from] : datetime  [to] : datetime  [room] : string  [place] : string  [special\_requirements] : string  [active] : int |  |
| edit\_excursion\_event\_book(ex\_ev\_book\_id, values) | ex\_ev\_book\_id : int  values : object ->  [visitors] : int |  |
| edit\_excursion\_event\_lecturer(ex\_event\_ id, values) | ex\_event\_id : int  values : object ->  [user\_id] : int |  |
| edit\_excursion\_event\_visitor(ex\_ev\_book\_id, values) | ex\_ev\_book\_id : int  values : object ->  [user\_id] : int  [visitors] : int |  |
| edit\_lecturer\_time(lec\_time\_id, values) | lec\_time\_id : int  values : object ->  [from] : datetime  [to] : datetime |  |
| edit\_page\_text(page\_name, values) | values : object ->  [page\_text] : text |  |
| edit\_payments(payment\_id, values) | [payment\_id] : int  values : object ->  [payment\_type] : int  [payment\_paid\_sum] : float  [categories][cat\_id] : int | Upravuje záznam v tabuľke *payments* z idčkom *payment\_id*. *Categories* je asoc. pole, ktoré treba prejsť cez *foreach*. V každej iterácií treba údaj z [categories][<prechadzany\_index>] použiť na úpravu záznamu z tabuľky fin\_redistribution , kde prechádzaný index identifikuje záznam podľa stĺpca *project\_category\_id*. V tomto zázname sa mení stĺpec *ratio*. |
| edit\_payments\_payment(payment\_id, values) | payment\_id : int  values : object ->  [payment\_id] : int  [payment\_user\_id] : int  [payment\_paid\_sum] : float  [categories][cat\_id] : int | v prípade, ak sa uhradí celá platba, treba uhradenú sumu pripočítať k jednotlivým kategóriám projektov podľa pomerov.  Treba z tabuľky *fin\_redistributes* vytiahnuť všetky záznamy týkajúce sa daného usera, podľa *ratio* prerozdeliť uhradenú sumu (payment\_paid\_sum) na jednotlivé *project\_category\_id*. |
| edit\_post(post\_id, values) | post\_id : int  values : object ->  [title] : string  [content] : string  [priority] : int  [post\_published] : tinyint |  |
| edit\_project(pr\_id, values) | pr\_id : int  values : object ->  [name] : string  [about] : string  [priority] : int  [project\_category\_ id] : int  [booked\_cash] : float  [from] : date  [to] : date |  |
| edit\_project\_category(pr\_cat\_id, values) | pr\_cat\_id : int  values : object ->  [project\_category\_name] : string |  |
| edit\_project\_category\_transactions(values) | [cash] : float  [from] : date |  |
| edit\_project\_closed(pr\_id) | pr\_id : int | funkcia nastaví príslušnému projektu (tabuľka *projects*) v stĺpci *project\_active* **false**. Okrem toho urobí sumár príslušných záznamov z tabuľky *project\_items* nad stĺpcom *project\_item\_price*, následne túto sumu odčíta od príslušného záznamu z *project\_categories* stĺpec *project\_category\_cash*. |
| edit\_project\_item(pr\_item\_id, values) | pr\_item\_id : int  values : object ->  [name] : string  [price] : float  [user\_id] : int  [Y-m-d] : date |  |
| edit\_study\_program(study\_pr\_id, values) | study\_pr\_id : int  values : object ->  [study\_program\_name] : string |  |
| edit\_user(user\_id, values) | user\_id : int  values : object ->  [username] : string  [name] : string  [surname] : string  [password] : string  [email] : string  [phone] : int  [study\_program\_id] : int  [degree\_id] : int  [place\_of\_birth] : string  [postcode] : int  [degree\_year] : int  [role] : int |  |

### Deleter

Tieto funkcie majú za úlohu zmazať údaje z databázy. Väčšinou sa problém zmazania týka nejednej tabuľky, nakoľko existujú medzi tabuľkami isté závislosti. Ku každej z nasledujúcich funkcií je v stĺpci „popis“ definovaný postup, akým spôsobom prebieha mazanie daného záznamu a ktorých ďalších tabuliek sa to týka.

|  |  |  |
| --- | --- | --- |
| Názov funkcie | Vstupy | Popis |
| remove\_degree(degree\_id) | degree\_id : int | zmaže záznam z tabuľky *degrees*, okrem toho nastaví všetkým príslušným záznamom v tabuľke *users* **null** |
| remove\_email\_type(e\_type\_id) | e\_type\_id : int | zmaže záznam z tabuľky *email\_types*, okrem toho nastaví všetkým príslušným záznamom v tabuľke *user\_email\_evidence* **null** |
| remove\_event(ev\_id) | ev\_id : int | zmaže záznam z tabuľky *events* |
| remove\_event\_category(ev\_cat\_id) | ev\_cat\_id : int | zmaže záznam z tabuľky *event\_categories*, okrem toho nastaví všetkým príslušným záznamom v tabuľke *events* **null** |
| remove\_excursion(ex\_id) | ex\_id : int | zmaže záznam z tabuľky *excursions*, okrem toho zmaže všetky príslušné záznamy z tabuľky *excursion\_events*, čím spustí rád ďalších operácií (viď funkciu *remove\_excursion\_event*) |
| remove\_excursion\_event(ex\_ev\_id) | ex\_ev\_id : int | zmaže záznam z tabuľky *excursion\_events*, okrem toho zmaže všetky príslušné záznamy z tabuľky *excursion\_times* a *booked\_excurions*. |
| remove\_excursion\_event\_book(ex\_ev\_book\_id) | ex\_ev\_book\_id : int | zmaže záznam z tabuľky *booked\_excursions* |
| remove\_excursion\_event\_lecturer(ex\_ev\_id, user\_id) | ex\_ev\_id: int  user\_id : int | zmaže všetky vyhovujúce záznamy z tabuľky *excursion\_times*, hodnoty oboch vstupných parametrov sa musia rovnať hodnotám príslušných stĺpcov, iba v tom prípade bude záznam vymazaný. |
| remove\_excursion\_event\_visitor(ex\_ev\_vi\_id) | ex\_ev\_book\_id : int | to isté ako *remove\_excursion\_event\_book* |
| remove\_expired\_users() |  | MY\_model |
| remove\_lecturer\_time(lec\_time\_id) | lec\_time\_id : int | zmaže záznam z tabuľky *excursion\_times* |
| remove\_payments(payment\_id) | payment\_id : int | zmaže záznam z tabuľky *payments*, ak existujú k nej záznamy vo *fin\_redistribution*, treba zmazať aj tie. |
| remove\_post(post\_id) | post\_id : int | zmaže záznam z tabuľky *posts*, okrem toho zmaže všetky príslušné záznamy z tabuľky *post\_modifiers*. |
| remove\_project(pr\_id) | pr\_id : int | zmaže záznam z tabuľky *projects*, okrem toho zmaže všetky príslušné záznamy z tabuľky *project\_items*. |
| remove\_project\_category(pr\_cat\_id) | pr\_cat\_id : int | zmaže záznam z tabuľky *project\_categories*, okrem toho nastaví všetkým príslušným záznamom v tabuľke *projects* **null** |
| remove\_project\_item(pr\_item\_id) | pr\_item\_id : int | zmaže záznam z tabuľky *project\_items* |
| remove\_study\_program(study\_pr\_id) | study\_pr\_id : int | zmaže záznam z tabuľky *study\_programs*, okrem toho nastaví všetkým príslušným záznamom v tabuľke *users* **null** |
| remove\_user(user\_id) | user\_id : int | zmaže záznam z tabuľky *users*, okrem toho zmaže všetky príslušné záznamy z tabuľky *payments*, *user\_email\_evidence*, *excursion\_times*, *booked\_excursions*. Nastaví hodnotu **null** príslušným stĺpcom v tabuľkách *post\_modifiers*, *posts (na stĺpci author\_id)*, *project\_items*, *projects*, *events (na stĺpci author\_id)*. |

## View

View sú podrobne definované v prílohe dokumentu. Je vytvorený grafický návrh jednotlivých stránok, ktorý má ilustrovať ich výzor. Tiež definuje komponenty formulárov. Tento grafický návrh však nezobrazuje presnú grafickú podobu stránky. Tá je ponechaná na programátora.

V návrhu sa vyskytujú výpisy hlavičiek databázových funkcií použitých na jednotlivých views. Oranžovou farbou sú vypísane funkcie, ktoré vracajú dáta. Väčšinou je táto funkcia graficky umiestnená pri komponente, ktorý má využiť jej navrátený obsah. Červenou farbou sú funkcie, ktoré realizujú zápis, úpravu alebo mazanie dát z databázy. Tieto sú graficky umiestnene pri konkrétnych view, ktorých sa týkajú.

Modrým podčiarknutým sú definované linky (odkazy) na controllery. Na výslednej stránke kliknutím na tieto linky sa spustí vykonávanie konkrétneho controllera.

## Controller

Tabuľka obsahuje zoznam controllerov a k nim príslušné modely a view ktoré používajú.

Nasledujúce URL adresy sú mapované nasledujúcim spôsobom:

doména/<názov\_controllera>/<názov\_metódy>/<param1>/<param2>/...

pričom *param1* a *param2* nie sú povinné a môže ich byť ľubovoľne. Ich existencia závisí od metódy controllera, či ich vyžaduje alebo nie.

V tabuľke sú použité nasledujúce notácie:

* <id> - v špicatých zátvorkách je označený premenlivý parameter, ktorý v adrese **musí** byť
* {id} - v množinových zátvorkách je označený voliteľný parameter, ktorý v adrese **môže**, ale **nemusí** byť

|  |  |  |
| --- | --- | --- |
| **URL** | **Modely** | **View** |
| /auth/index |  | auth\_view |
| /auth/login |  | - |
| /auth/logout |  | - |
| /auth/registration | get\_study\_programs()  get\_degrees()  add\_register() | auth\_registration |
| /correspondence/index | get\_email\_types()  get\_study\_programs() | correspondence\_view |
| /correspondence/review | get\_users(study\_ids, degrees, degree\_years) | correspondence\_review\_view (grid) |
| /correspondence/send\_email | add\_email\_log(email\_types, user\_ids) | message |
| /degrees/add | add\_degree(values) | degrees\_view (grid) |
| /degrees/delete/<id> | remove\_degree(degree\_id) | message |
| /degrees/edit/<id> | edit\_degree(degree\_id, values) | degrees\_view (grid) |
| /degrees/index | get\_degrees() | degrees\_view (grid) |
| /email\_types/add | add\_email\_type(values) | email\_types\_view (grid) |
| /email\_types/delete/<id> | remove\_email\_type(e\_type\_id) | message |
| /email\_types/edit/<id> | edit\_email\_type(e\_type\_id, values) | email\_types\_view (grid) |
| /email\_types/index | get\_email\_types() | email\_types\_view (grid) |
| /event\_categories/add | add\_event\_category(values) | event\_categories\_view (grid) |
| /event\_categories/delete/<id> | remove\_event\_category(ev\_cat\_id) | massage |
| /event\_categories/edit/<id> | edit\_event\_category(ev\_cat\_id, values) | event\_categories\_view (grid) |
| /event\_categories/index | get\_event\_categories() | event\_categories\_view (grid) |
| /events/add | get\_event\_categories()  add\_event(values) | events\_add\_view |
| /events/delete/<id> | remove\_event(ev\_id) | confirm\_view |
| /events/detail/<id> | get\_event\_detail(event\_id) | events\_detail\_view |
| /events/edit/<id> | get\_event\_detail(event\_id)  edit\_event(ev\_id, values) | events\_edit\_view |
| /events/index/{id} | get\_events(cat\_id) | events\_view |
| /events/newest/{id} | get\_events\_newest(cat\_id) | events\_view |
| /events/prior/{id} | get\_events\_prior(cat\_id) | events\_view |
| /excursion\_events/add/<ex\_id> | get\_lecturer\_times(ex\_event\_id, user\_id)  add\_excursion\_event(values) | excursion\_events\_add\_view |
| /excursion\_events/book/<ex\_id> | get\_excursion\_events(excursion\_id)  get\_excursion\_event\_free(excursion\_event\_id)  add\_excursion\_event\_book(values) | excursion\_events\_book\_view |
| /excursion\_events/book\_delete/<id> | remove\_excursion\_event\_visitor(ex\_ev\_vi\_id) | message |
| /excursion\_events/book\_edit/<id> | get\_excursion\_event\_book(ex\_ev\_book\_id)  get\_excursion\_event\_free(excursion\_event\_id)  edit\_excursion\_event\_book(ex\_ev\_book\_id, values) | excursion\_events\_book\_edit\_view |
| /excursion\_events/delete/id | remove\_excursion\_event(ex\_ev\_id) | message |
| /excursion\_events/detail/<id> | get\_excursion\_event\_detail(ex\_event\_id)  get\_excursion\_event\_lecturers(ex\_event\_id)  get\_excursion\_event\_visitors(ex\_event\_id)  get\_users(role) | excursion\_events\_detail\_view |
| /excursion\_events/edit/<id> | get\_lecturer\_times(ex\_event\_id, user\_id)  get\_excursion\_event\_detail(ex\_event\_id)  edit\_excursion\_event(ex\_ev\_id, values)  add\_excursion\_event\_visitor(values)  edit\_excursion\_event\_visitor(ex\_ev\_book\_id, values)  remove\_excursion\_event\_visitor(ex\_ev\_vi\_id) | excursion\_events\_edit\_view |
| /excursion\_events/lecturer\_times/<user\_id> | get\_lecturer\_times(ex\_event\_id, user\_id)  add\_lecturer\_times(ex\_event\_id, user\_id, values)  edit\_lecturer\_time(lec\_time\_id, values)  remove\_lecturer\_time(lec\_time\_id, values) | excursion\_events\_lecturer\_times\_view |
| /excursion\_events/lecturers/{ex\_ev\_id} | get\_excursion\_event\_lecturers(ex\_event\_id)  get\_users(role)  add\_excursion\_event\_lecturer(ex\_event\_id, values)  edit\_excursion\_event\_lecturer(ex\_event\_ id, values)  remove\_excursion\_event\_lecturer(ex\_ev\_id, user\_id) | excursion\_events\_lecturers\_view |
| /excursions/add | get\_excursion\_detail(excursion\_id)  php – generate priority  add\_excursion(values) | excursions\_add\_view |
| /excursions/delete/<id> | remove\_excursion(ex\_id) | message |
| /excursions/detail/<id> | get\_excursion\_detail(excursion\_id) | excursions\_detail\_view |
| /excursions/edit/<id> | get\_excursion\_detail(excursion\_id)  php – generate priority  edit\_excursion(ex\_id, values) | excursions\_edit\_view |
| /excursions/index | get\_excursions() |  |
| /excursions/index/<from>/<to> | get\_excursions(from, to) | excursions\_view |
| /io/export |  | io\_export\_view |
| /io/import |  | io\_import\_view |
| /payments/delete/<id> | remove\_payments(payment\_id) | message |
| /payments/edit/<id> | edit\_payments(payment\_id, values) | payments\_view (grid) |
| /payments/index/{id} | get\_users(role)  get\_payments(user\_id) | payments\_view (grid) |
| /payments/nopaid/{id} | get\_users(role)  get\_payments\_nopaid(user\_id) | payments\_view (grid) |
| /payments/paid/{id} | get\_users(role)  get\_payments\_paid(user\_id) | payments\_view (grid) |
| /payments/add | add\_payments(values) | payments\_view (grid) |
| /posts/add | php – generate priority  add\_post(values) | posts\_add\_view |
| /posts/delete/<id> | remove\_post(post\_id) | message |
| /posts/detail/<id> | get\_post\_detail(post\_id) | posts\_detail\_view |
| /posts/edit/<id> | php – generate priority  get\_post\_detail(post\_id)  edit\_post(post\_id, values) | posts\_edit\_view |
| /posts/index | get\_posts() | posts\_view |
| /posts/modifiers/<id> | get\_post\_modifiers(post\_id) | posts\_modifiers\_view |
| /project\_categories/add | add\_project\_category(values) | project\_categories\_view (grid) |
| /project\_categories/delete/<id> | remove\_project\_category(pr\_cat\_id) | message |
| /project\_categories/detail/<id> | get\_category\_detail(cat\_id)  get\_event\_categories()  get\_transactions(pr\_cat\_id)  get\_projects(cat\_id) | project\_categories\_detail\_view |
| /project\_categories/edit/<id> | edit\_project\_category(pr\_cat\_id, values) | project\_categories\_view (grid) |
| /project\_categories/index | get\_project\_categories\_total\_cash()  get\_project\_categories() | project\_categories\_view (grid) |
| /projects/add | get\_project\_categories() | project\_add\_view |
| /projects/delete/<id> | remove\_project(pr\_id) | message |
| /projects/detail/<id> | get\_project\_detail(project\_id)  get\_project\_items(project\_id) | projects\_detail\_view |
| /projects/edit/id | get\_project\_detail(project\_id)  get\_project\_categories()  get\_project\_items(project\_id)  edit\_project(pr\_id, values) | projects\_edit\_view |
| /projects/index | get\_project\_categories()  get\_projects(cat\_id) | projects\_view |
| /project/add\_project\_item/<id> | add\_project\_item(pr\_id, values) | projects\_edit\_view (grid) |
| /project/edit\_project\_item/<id> | edit\_project\_item(pr\_item\_id, values) | projects\_edit\_view (grid) |
| /project/delete\_project\_item/<id> | remove\_project\_item(pr\_item\_id) | projects\_edit\_view (grid) |
| /project/edit\_project\_close/<id> | edit\_project\_closed(pr\_id) | projects\_edit\_view (grid) |
| /studies/add | add\_study\_program(values) | studies\_view (grid) |
| /studies/delete/<id> | remove\_study\_program(study\_pr\_id, values) | studies\_view (grid) |
| /studies/edit/<id> | edit\_study\_program(study\_pr\_id, values) | studies\_view (grid) |
| /studies/index | get\_study\_programs() | studies\_view (grid) |
| /users/add | get\_study\_programs()  get\_degrees()  get\_project\_categories()  add\_user(values) | user\_add\_view |
| /users/delete/<id> | remove\_user(user\_id) | message |
| /users/detail/<id> | get\_user\_detail(user\_id)  get\_payments\_lastpaid(user\_id)  add\_payments(values)  remove\_payments(payment\_id) | users\_detail\_view |
| /users/edit/<id> | get\_user\_detail(user\_id)  get\_study\_programs()  get\_degrees()  get\_project\_categories()  get\_fin\_redistribution(user\_id, pr\_cat\_id)  edit\_user(user\_id, values) | users\_edit\_view |
| /users/index | get\_users(role) | users\_view |
| /users/members | get\_users(role)  get\_payments\_nopaid(user\_id) | users\_view |
| /users/admins | get\_users(role) | users\_view |
| /users/potentials | get\_users(role) | users\_view |
| /users/innactive | get\_users(role) | users\_view |
| /users/blocked | get\_users(role) | users\_view |

Grafické znázornenie viď. príloha str.č.1

# Motivácia

