# Triedny Návrh

## Modely

Umožňujú prácu s databázou využitím ORM, ktoré poskytuje Nette. Každá tabuľka má svoj model (resp. repository).

### UserRepository

Slúži pre vyberanie a vkladanie a mazanie dát pre tabuľku User.

**Metódy**

* getUserByEmail($email): vráti záznam o užívateľovi podľa zadaného parametra $email
* getUserById($id): vráti záznam o užívateľovi podľa id užívateľa
* getUserTimeCreated($id): vráti čas, kedy bol užívateľ s daným id vytvorený
* addUser($user\_array): pridá nového užívateľa do databázy
* removeUser($user\_id): odstráni užívateľa z databázy a aj jeho príklady (foreign key v Task a Test tabuľkách)
* updateUserPassword($user\_id, $new\_password, $hash): zmení heslo užívateľovi
* getUserByHash($hash): vráti užívateľa, ktorý si chce zmeniť heslo – podľa prideleného $hash
* generateHashForUser($hash): pridelí hash užívateľovi, ktorý si chce zmeniť heslo
* removeStudentFromGroup($group\_id, $student\_id): odstráni študenta z danej skupiny
* getStudentsByGroup($group\_id): vráti záznamy o študentoch z danej skupiny z tabuľky

### GroupRepository

Spravuje tabuľku group, kde sú záznamy o učiteľom vytvorených skupinách

**Metódy**

* addGroup($teacher\_id, $group\_name, $group\_key, $description): pridá skupinu do tabuľky
* removeGroup($group\_id): odstráni skupinu z tabuľky (a tým kaskádovito aj žiakov a ich príklady)
* getTeacherGroups($teacher\_id): vráti záznamy o skupinách daného učiteľa

### TaskRepository

Spravuje údaje o vyriešených príkladoch študentov

**Metódy**

* addTask($id\_user, $id\_unit, $id\_test, $convert\_array, $isTest=false): pridá príklad do tabuľky. Pole $convert\_array bude mať info o príklade (nb\_value\_from, nb\_power\_from, nb\_value\_to, nb\_power\_to)
* markTaskAsCorrect($id\_task, $is\_correct): označí v tabuľke, či žiak správne vyplnil úlohu
* getTasksByStudent($student\_id): vráti príklady na základe id študenta
* getTasksBytTest($test\_id): vráti dvojrozmerné pole, kde prvý rozmer budú študenti a druhý ich riešené príklady

### UnitRepository

Slúži na komunikáciu s tabuľkou unit, kde sú informácie o fyzikálnych jednotkách

**Metódy**

* getUnitById($unit\_id): vráti záznam o fyz. jednotke podľa id jednotky
* getBaseUnitByDifficulty($difficulty): vráti náhodne záznam o základnej jednotke podľa obtiažnosti
* getRandomUnitForBaseUnit($unit\_id): vráti náhodnú jednotku ku základnej jednotke (okrem nej) z tej istej veličiny

### TestRepository

Pracuje s tabuľkou, kde sú uložené informácie o testoch pre študentov

**Metódy**

* addTest($id\_group, $count, $difficulty=1): pridá nový test
* closeTest($test\_id): ukončí daný test

## Controllery

Controllery MVC frameworku Nette slúžia na komunikáciu s Modelmi (cez ktoré dopytujú dáta z databázy a modifikujú ich). Tieto dáta môžu ďalej spracovávať a poskytnúť ich View-om, ktoré slúžia ako HTML šablóny. Ďalej spracovávajú dáta získané z formulárov. Každý Controller dedí z triedy BasePresenter.

### Auth Controller

Riadi autentifikáciu používateľa (študenta/učiteľa) – registráciu, prihlasovanie, odhlasovanie, zmenu hesla používateľa. Tento controller komunikuje s triedou \Nette\Security\User, ktorá uchováva informáciu o skupine, užívateľovi a obsahuje metódy na prácu s užívateľom – registrácia, prihlásenie, odhlásenie, zmenu hesla v databáze.

**Metódy**

* createComponentNewRegisterUser(): vráti formulár na registráciu užívateľa (žiaka, alebo učiteľa)
* newRegisterUserSubmitted(): zvaliduje a spracuje odoslaný registračný formulár pomocou triedy \Nette\Security\User
* createComponentNewLoginForm(): vráti formulár na prihlásenie užívateľa (žiaka, alebo učiteľa)
* newLoginFormSubmitted(): spracuje odoslaný prihlasovací formulár pomocou triedy \Nette\Security\User. Z tejto triedy sa zavolá metóda login($user, $password), kde sa pomocou parametrov pošle email a heslo zadané užívateľom vo formulári. Tá prihlási užívateľa nastavením Sessions, kde si pamätá informácie o užívateľovi.
* newLogoutFormSubmitted(): metóda odhlasuje prihláseného užívateľa

### Student Controller

Riadi požiadavky od prihláseného študenta. Pri každej z týchto akcií musí byť študent prihlásený. To sa overuje v metóde Startup() tejto triedy.

**Metódy**

* actionDefault(): zobrazí študentovi jeho domovskú stránku po prihlásení
* actionSolutions(): spracuje a zobrazí študentovi jeho riešené úlohy
* createComponentNewTask(): vygeneruje žiakovi príklady pomocou triedy UnitConversion, uloží do databázy prostredníctvom modelu a zobrazí mu ich na riešenie
* createComponentNewTest(): vygeneruje žiakovi príklady z danej päťminutovky podľa nastavenej obtiažnosti pomocou triedy UnitConversion a uloží do databázy. Príklady na riešenie zobrazí študentovi
* newTaskSubmitted(isTest = false): vyhodnotí odoslané príklady žiaka a uloží jeho výsledky do databázy pomocou modelu a presmeruje ho na actionShowResult()
* actionShowResult(): zobrazí študentovi výsledky (ktoré príklady mal dobre/zle, alebo nerieš)

### Teacher Controller

Riadi požiadavky od prihláseného učiteľa. Pri každej z týchto akcií musí byť učiteľ prihlásený. Pri každej z týchto akcií musí byť študent prihlásený.

**Metódy**

* actionDefault (): zobrazí učiteľovi jeho domovskú stránku po prihlásení. Tu mu vylistuje jeho pridané skupiny a možnosť vytvoriť nové skupiny.
* createComponentNewGroup(): vráti formulár na vytvorenie skupiny.
* newGroupSubmitted(): vytvorí novú skupinu zo zadaných údajov vo formulári a presmeruje na actionDefault()
* actionRemoveGroup($group\_id): odstráni učiteľovu skupinu a s ňou jej študentov a ich príklady
* actionShowStudentsInGroup($group\_id): zobrazí učiteľovi študentov z danej skupiny
* actionShowStudentsResults($student\_id, $only\_tests=false): zobrazí učiteľovi študentove riešené príklady
* actionSetTest($count, $difficulty): učiteľ zadá test pre žiakov (prostredníctvom modelu sa nastaví v db., že daná skupina má test)
* actionCloseTest(): učiteľ ukončí testovanie (skončí sa v DB) a presmeruje na actionShowTestResults()
* actionShowTestResults(): zobrazia sa učiteľovi výsledky žiakov z ukončenej päťminútovky
* actionShowGroupStatistics(): vráti štatistiky danej skupiny v podobe grafov vyšpecifikovaných v predošlých dokumentoch

## Knižnice, samostatné triedy

### UnitConversion

Táto trieda bude slúžiť na generovanie príkladov na premeny jednotiek a ich kontrolu. Bude pri tom využívať potrebné modely na prácu s databázou.

* generateConversion($difficulty = null, $unit = null): metóda vygeneruje a vráti príklad v tvare [‘id\_phys\_quantity’=>1, ‘from\_unit’=>’centimeter’, ‘from\_unit\_symbol’=>’cm’, ‘to\_unit’=>’meter’, ’to\_unit\_symbol’=>’m’, ‘value’=>166.3]
* generateManyConversions($count, $difficulty = null, $unit = null): vygeneruje taký počet príkladov, ako sa nastaví parameter $count. Metóda vráti tieto príklady ako pole, ktorého prvky budú mať rovnaký tvar, aký vracia metóda generate(…)
* checkConversion($id\_phys\_quantity, $from\_unit, $value, $result\_to\_check): skontroluje, či je správne premenená jednotka z $from\_unit na základnú jednotku. Metóda vracia true/false.
* getBaseFormatNumberHint($number): metóda vráti základný tvar čísla. Napríklad 166.3 vráti 1.66, alebo z 0.017 vráti 1.7
* getExp10Hint($id\_phys\_quantity, $from\_unit, $value): metóda vracia pomôcku na výsledok prevodu jednotiek v tvare mocniny 10. Napr. 0.003dm=[…]\*10^**-4**m