UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Nový dlhodobý viacúčelový sklad úloh na cvičenia

Návrh systému

Bratislava, 2013

Andrej Jursa

Obsah

1 Úvod

Tento dokument obsahuje návrh nového systému LIST (Long-term Internet Storage of Tasks). Na vytvorenie nového systému bude použitých niekoľko open-source riešení, ktoré majú dostatočne dobre napísanú dokumentáciu. Ich detailný popis je možné nájsť v samotnom dokumente bakalárskej práce.

Týmito použitými technológiami budú:

- CodeIgniter php application framework,
- Smarty php template engine,
- DataMapper ORM pre CodeIgniter,
- GeSHi php general syntax highlighter,
- jQuery + jQuery-UI,
- Fancybox,
- TinyMCE a Plupload.

Pre implementáciu je možné použitie aj ďalších open-source riešení menších formátov.

1.1 Prehľad o obsahu dokumentu

Tento dokument obsahuje konceptuálnu analýzu nového systému, špecifikáciu komponentov a architektúru niektorých pridaných komponentov.

2 Požiadavky na softvér

V tejto kapitole sformujeme požiadavky na softvérový systém L.I.S.T., ktoré sú nasledovné:

- systém musí podporovať paralelne bežiace výukové kurzy ako aj umožniť paralelné hodnotenie odovzdaných riešení,
- systém bude používaný dvoma úrovňami používateľov, učiteľom a študentom. Tieto dve úrovne musia byť dve nezávislé entity a každá táto entita bude mať svoje vlastné rozhranie, v ktorom bude pracovať,
- systém musí umožniť vytváranie a správu úloh a príloh k úlohám (prílohy vo forme súborov uložených sa serveri), je tiež nutné, aby boli úlohy kategorizované a aby sa dali filtrovať podľa týchto kategórií,
- kategórie úloh musia mať hierarchickú štruktúru,
- systém musí umožňovať vytvárať zostavy úloh, alebo inak povedané zadania pre študentov, ktoré budú pozostávať z viacerých úloh, tak isto jedna úloha môže patriť k viacerým zadaniam,
- systém by mal umožniť klonovať úlohy aj zadania úloh, (napríklad kvôli zmenám, ktoré chceme aby sa prejavili lokálne),
- systém by mal podporovať viacjazyčné používateľské rozhranie a mal by umožňovať prekladať obsah (úlohy, zostavy atď …) do viacerých jazykov,
- systém by mal umožňovať vytvárať výučbové kurzy, ku kurzom vytvárať skupiny a ku skupinám priraďovať miestnosti, v ktorých prebieha výučba pre danú skupinu,
- systém musí umožniť študentovi prihlásiť sa na kurz a vybrať si študijnú skupinu, učiteľovi musí umožniť spravovať priradenie študentov do skupín a riadiť ich účasť v kurzoch,
- systém by mal študentom dovoliť odovzdať svoje riešenie zverejnených zadaní (zostáv úloh) online a učiteľovi by mal umožniť tieto úlohy kontrolovať a hodnotiť,
- systém by mal umožňovať vytvárať k jednotlivým úlohám unit testy, vygenerovať z nich spustiteľný kód a otestovať nimi odovzdané riešenie so zobrazením výsledkov.

3 Konceptuálna analýza

3.1 Používatelia systému

Softvér umožní pracovať so systémom dvom úrovniam používateľov, študentom a učiteľom. Každá úroveň používateľa bude mať na prácu so systémom vyhradenú vlastnú časť systému.

Študent bude vystupovať v roli bežného používateľa, ktorí má minimálne možnosti, ohraničené na prihlasovanie sa do študentského rozhrania, prihlasovanie sa na kurzy, výber skupiny v kurzoch, prezeranie zadaní úloh, odovzdávanie vlastného riešenia, prezeranie hodnotenia odovzdaných riešení a úpravu vlastného používateľského účtu.

Učiteľ bude vystupovať v roli administrátora systému, a bude sa vedieť so svojím účtom prihlasovať do administrácie, kde bude vedieť meniť takmer všetky parametre obsahu. Všetci učitelia sú si rovní, všetci budú mať prístup ku tým istým možnostiam správy systému.

Používateľské účty budú mať minimálne nastavenia, ako prihlasovacie meno sa bude používať e-mailová adresa. Ak bude učiteľ chcieť, môže si vytvoriť aj študentský účet s rovnakou e-mailovou adresou ako používa na učiteľskom účte, avšak vrámci tabuliek študentov a učiteľov budú e-mailové adresy unikátne.

3.2 Jazykový preklad systému

Systém má byť vytvorený tak, aby umožňoval preklad rozhrania a obsahu do viacerých jazykov. Za týmto účelom budú v systéme prítomné tri riešenia, ako tohto efektu dosiahnuť.

Prvé riešenie bude predpokladať existenciu súborov s definíciou textu v danom jazyku priradenou k nejakému kľúčovému slovu – konštante. Toto riešenie je orientované hlavne na preklad používateľského rozhrania. Takýto textový súbor sa načíta a páry konštanta – text sa uložia do spoločného zásobníka, odkiaľ sa budú dať získať pre potreby zobrazenia na stránke systému.

Druhé riešenie spočíva v existencii tabuľky prekladov v databáze, ktorá bude obsahovať informácie jednak o zvolenom názve konštanty, tak aj o jazyku, ktorý je ňou pokrytý a konečne text v danom jazyku. Pre každý takýto preklad sa pripojí prefix ku konštante a všetky takto poopravené páry konštanta – text sa vložia do rovnakého zásobníka ako ho rieši prvé riešenie, čo znamená, že prístup k takémuto prekladu bude rovnaký ako v prvom riešení.

Tretie riešenie bude počítať s existenciou inej tabuľky, ktorá bude realizovať prekrytie pôvodného textu textom v inom jazyku. Všetok bežný textový obsah sa bude ukladať akoby v nedefinovanom východzom jazyku a k vybraným položkám bude možné definovať prekrytie tohto textu v konkrétnom jazyku. Tabuľka týchto prekrytí bude obsahovať informáciu o texte prekrytia,

ktorý spadá do nejakej konkrétnej tabuľky, špecifikovanej hodnotou jej ID, informáciou o tom, v ktorom stĺpci tejto tabuľky má byť text prekrytý a o aký jazyk sa jedná. Preklad realizovaný pomocou prekrytia bude následne fungovať tak, že sa z pôvodnej tabuľky načíta záznam a potom sa načítajú príslušné texty prekrytí k príslušným stĺpcom. Tieto prekrytia je nutné realizovať optimálne, aby bolo možné pre viac záznamov jedným volaním databázového dopytu inicializovať všetky potrebné prekrytia pre konkrétny jazyk a tabuľku.

Druhé a tretie riešenie sa môže dopĺňať. Z návrhu vyplýva, že sa môže aj kombinovať, teda v prekrytí by mohla byť definovaná konštanta, ktorej text sa má zobraziť, avšak toto správanie je z pohľadu celkovej funkčnosti nevyužiteľné. Teda v systéme sa bude používať v konkrétnej tabuľke a jej konkrétnom stĺpci preklad buď pomocou druhého alebo tretieho riešenia.

3.3 Úlohy a zostavy úloh

Systém ma umožniť vytvárať, upravovať a uchovávať čiastkové úlohy (ďalej len úlohy), ktoré pozostávajú z názvu úlohy, textu úlohy, jej kategorizácie vrámci editovateľnej hierarchickej štruktúry kategórií a súborov k úlohe.

Každá úloha musí implementovať jazykové prekrytie, kvôli lokalizácii vlastného textu zadania do viacerých jazykov, ktoré systém podporuje.

Úloha môže aj nemusí mať vzorové riešenia, tie však nebudú viditeľné pre študentov, budú slúžiť len učiteľom.

Samotné úlohy nie sú viditeľné pre študentov. Na to aby bola niektorá úloha zobrazená študentovi, musí byť zaradená do zostavy úloh (v tomto texte tiež označované ako zadanie). Zostava úloh je kombináciou rôznych úloh, ktorú bude zostavovať učiteľ a ako taká bude aj určovať ktorým študentom sa zobrazí a kedy.

Na nastavenie zobrazovania bude mať zostava niekoľko volieb. Každá zostava bude musieť byť zaradená do niektorého vyučovacieho kurzu a bude musieť byť označená nejakým typom zostavy (ten bude aj hovoriť o tom, či študent bude alebo nebude odovzdávať súbory s riešením pre túto zostavu).

Voliteľne sa bude dať nastaviť skupina, v rámci zvoleného kurzu, len pre ktorú sa má zostava zobrazovať.

Bude možné zadať časovú informáciu, dátum a čas, kedy sa má začať zostava zobrazovať. Rovnakým spôsobom sa bude dať nastaviť dátum a čas ukončenia odosielania súborov s riešením. Ak tieto časové informácie nebudú zadané, zostava sa jednoducho zobrazí ihneď a odosielanie súborov nebude nijako časovo limitované.

Ak to bude potrebné, bude môcť učiteľ zvoliť miestnosť v ktorej prebieha vyučovanie predtým

zvolenej skupiny, čo spôsobí, že sa zostava zobrazí v momente, keď začne vyučovanie v čase definovanom v miestnosti. Tu sa bude brať do úvahy čas začiatku publikovania, ktorým sa vymedzí čas, časový bod, od ktorého sa má prihliadať na splnenie času a dňa v týždni definovanom v miestnosti. To znamená, že ak má skupina miestnosť, napríklad H3, v ktorej začína tejto skupine výučba v utorok o 9:50, túto miestnosť učiteľ zvolí a ako čas začiatku publikovania nastaví 22.9.2014 00:00:00, pondelok ráno, k zverejneniu zostavy príde v utorok 23.9.2014 o 09:50:00. Samozrejme, že ak nebude nastavený žiaden čas zverejnenia zostavy, výber miestnosti nijako neovplyvní, kedy sa zostava zobrazí.

Ak zostava raz splní podmienku svojho zobrazenia, ostane zobrazená nastálo. Bude tu však ešte jedna možnosť a to prepínač, ktorý bude hovoriť, či je alebo nie je možné zostavu publikovať. Ak tento prepínač učiteľ vypne, aj predtým zverejnená zostava sa zverejňovať prestane.

Keďže je občas nutné pozmeniť zadanie úlohy alebo tú istú zostavu zadať napríklad inej skupine, bude možné zostavy aj úlohy klonovať. To spôsobí, že sa celý obsah a súbory pôvodnej úlohy skopírujú a uložia ako nová úloha a v prípade klonovania zostavy sa vytvorí nová zostava s rovnakými úlohami (skopíruje sa zostava a relácie k úlohám), ale v kópii zostavy sa nastaví automaticky jej nezverejňovanie. Učiteľ tak bude môcť bezpečne nastaviť zverejňovanie kópie a študent neuvidí duplikát zostavy v svojom zozname zadaní.

3.4 Unit testy

K jednotlivým úlohám bude možné vytvoriť unit testy. Na vytváranie unit testov bude existovať abstraktný adaptér a z neho odvodené inštancie pre unit testy pre rôzne programovacie jazyky.

Abstraktný adaptér bude viac-menej interface, ktorý budú od neho odvodené triedy pre jednotlivé programovacie jazyky implementovať. Takéto riešenie by malo zjednotiť správu unit testov, keďže pre všetky programovacie jazyky bude návrh takmer rovnaký.

Spoločným menovateľom adaptérov pre unit testy bude podobný editor unit testov v administrácii, kompletizácia unit testov do skompilovateľného kódu a spúšťanie unit testov pre jednotlivé súbory riešení.

Samotný návrh editora pozostáva z troch tabuliek v databáze.

Prvá tabuľka unit testov bude popisovať ku ktorej úlohe unit testy patria a aký programovací jazyk (adaptér unit testov) používajú.

Druhá tabuľka bude obsahovať informácie o menách jednotlivých test case a bude odkazovať na prvú tabuľku. Teda unit testy pre jednu úlohu budú môcť mať viacero test case.

Tretia tabuľka bude popisovať testovacie funkcie vrámci test case. Bude obsahovať odkaz na druhú tabuľku, bude mať unikátne meno testovacej funkcie vrámci test case, kód samotnej

funkcie a typ tejto funkcie (napríklad anotácia funkcie).

K jednotlivým unit testom ako celku bude možné pridávať súbory, či už dátové súbory potrebné pre testy alebo súbory s zdrojovým kódom, ktoré sú vyžadované pri kompilácii testov.

Keďže unit testy budú viazané na samostatné úlohy, pri testovaní zostavy úloh sa budú vždy spúšťať pre každú jednu členskú úlohu zostavy samostatne s tým istým študentským riešením.

Z rovnakých dôvodov náväznosti na samostatné úlohy sa budú aj unit testy klonovať spolu s úlohou (ak sa učiteľ rozhodne klonovať úlohu, ktorá obsahuje aj unit testy).

3.5 Zostavovanie unit testov

Zostavovanie unit testov bude riešené odvodeným adaptérom pre unit testy pre každý programovací jazyk samostatne, ale mal by zohľadňovať tieto princípy:

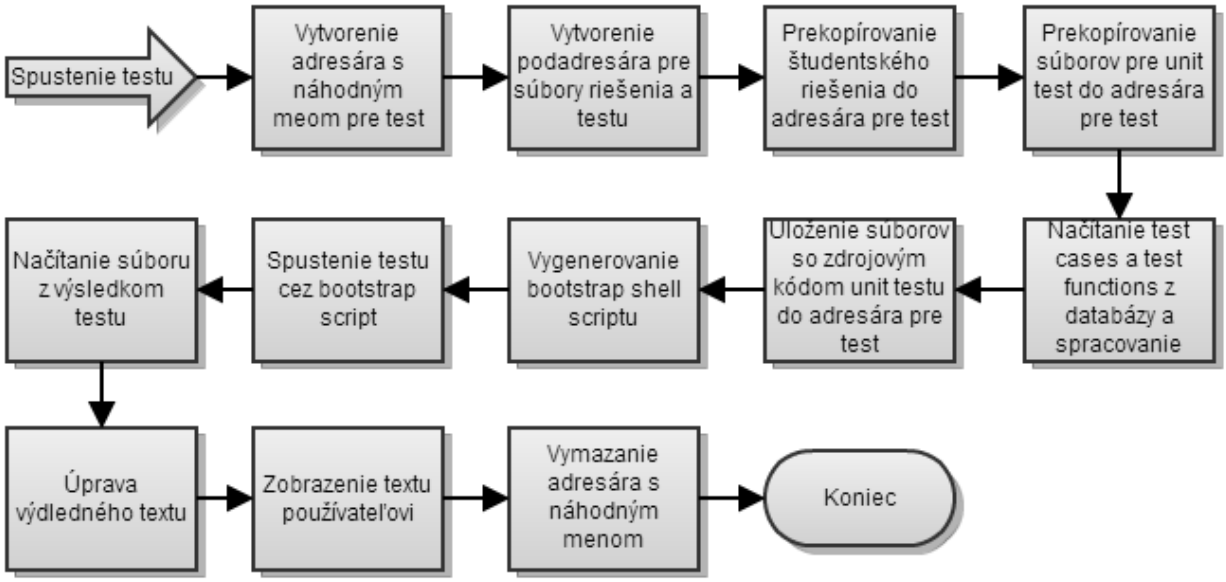
1. pred kompiláciou unit testov sa vytvorí adresár s náhodným menom, v ktorom sa vytvorí podadresár pre zostavované unit testy,
2. do podadresára sa rozbalí študentské riešenie zadania,
3. do podadresára sa skopírujú dodatočné súbory k testom, ak je treba, prepíšu existujúce súbory,
4. načítajú sa test_cases a test_functions, vygeneruje sa obsah súborov s testami a uloží sa do podadresára,
5. v nadradenom adresári s náhodným menom sa vytvorí bootstrap shell script, ktorý bude spúšťať test.

3.6 Spustenie unit testu

Po zostavení testu a vytvorení bootstrap shell scriptu spustí tento shell script. Počas behu skriptu by mal byť vygenerovaný súbor s výstupom testovania, ktorý následne adaptér načíta. Adaptér môže voliteľne implementovať lexikálny analyzátor, ktorý môže nejak pozmeniť text načítaný so súboru (napr. správy o chybách označí červenou farbou, aby sa ľahšie identifikovali). Následne sa text zobrazí používateľovi, ktorý spustil test.

Keďže študent bude spúšťať test celkovo pre celú zostavu, načíta sa zoznam unit testov pre každú úlohu, v náhodnom poradí, a postupne sa každý test vybuduje a spustí.

Po dokončení testu sa vymaže príslušný adresár, v ktorom sa nachádzajú súbory a bootstrap shell script k testu.



Obrázok 1: Postup vygenerovania a spustenia unit testov.

3.7 Zálohovanie a obnova systému

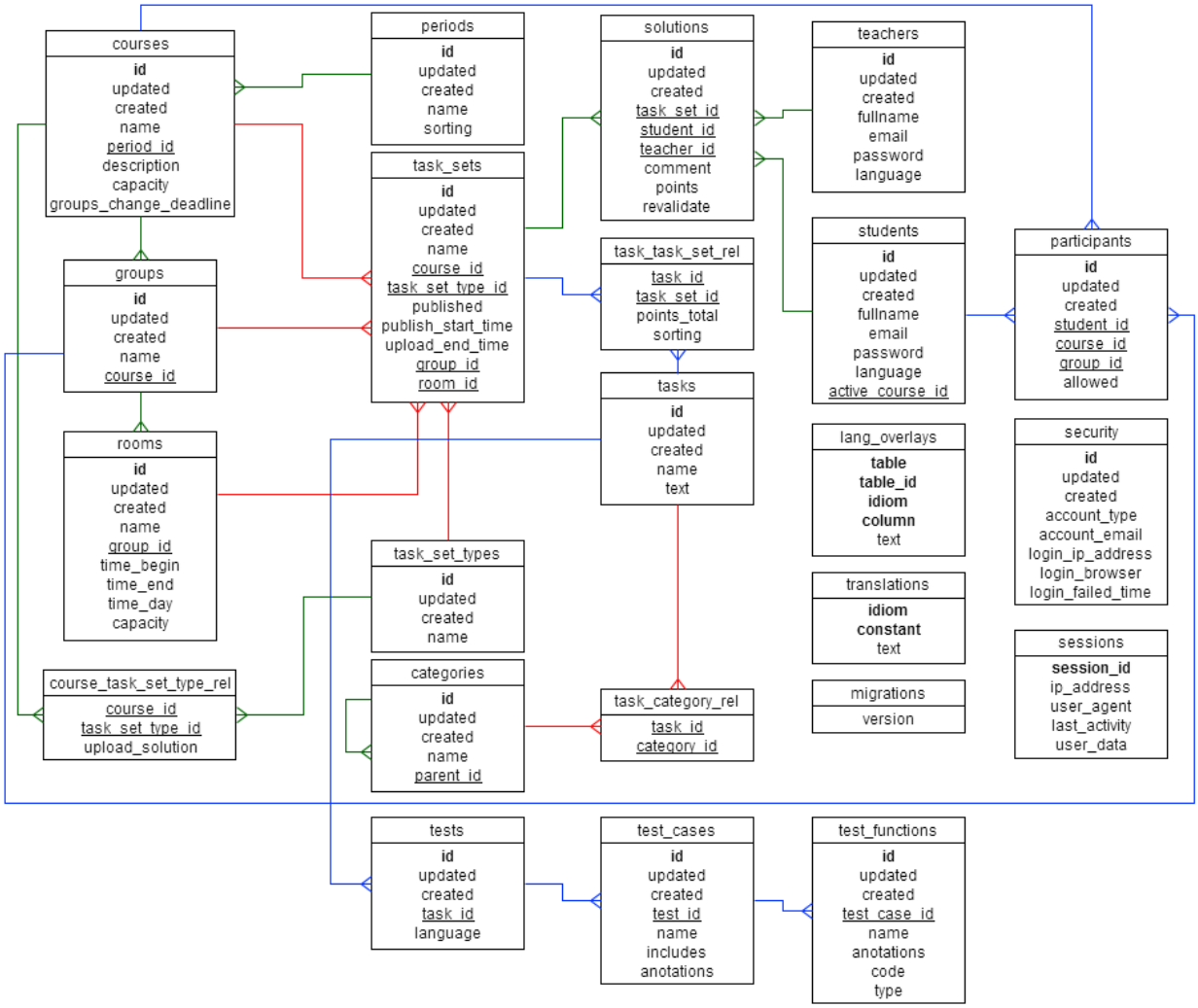
Keďže systém má dlhodobo skladovať úlohy, musí byť schopný dáta o úlohách zálohovať a obnovovať.

Aby bolo možné toto dosiahnuť, bude systém obsahovať zálohovací podsystém, ktorý bude prístupný z príkazového riadku. Pred spustením zálohy sa zamkne zbytok systému, aby nebolo možné meniť dáta z administrácie alebo študentského rozhrania, čiže záloha bude konzistentná. Zálohované budú všetky súbory a databáza a výsledok bude uložený v ZIP archíve. Po dokončení a zapísaní archívu na disk bude systém odblokovaný.

Obnovovanie systému bude fungovať rovnako z príkazového riadku. Pri obnovovaní systému sa zadá zdrojový ZIP archív, zamkne sa systém a obnovia sa všetky súbory do pôvodného stavu a tiež databáza. Obnova zmaže pôvodné súbory a celú databázovú štruktúru.

Riešenie zálohovania pomocou príkazového riadku dovolí vykonávať zálohu pomocou CRON skriptu na serveri.

3.8 Dátová štruktúra databázy



Obrázok 2: Databázová štruktúra

V zjednodušenej schéme zapojenia databázových tabuliek do relácií platia tieto pravidlá:

- na hrubo označený text je časťou primárneho kľúča tabuľky,
- podčiarknutý text je cudzím kľúčom do inej tabuľky,
- názvy tabuliek končiace sufixom _rel označujú JOIN tabuľky, nebudú v zdrojovom kóde inštanciované ako objekty modelov.

Päť tabuliek, migrations, translations, lang_overlays, security a sessions sú pomocné tabuľky a tabuľky obsahujúce jazykové preklady, nie sú v žiadnej priamej relácii s inými tabuľkami.

Táto štruktúra databázy je predmetom zmien vo vývojovom procese a nie je finálnou štruktúrou.

3.9 Dátová štruktúra súborového systému

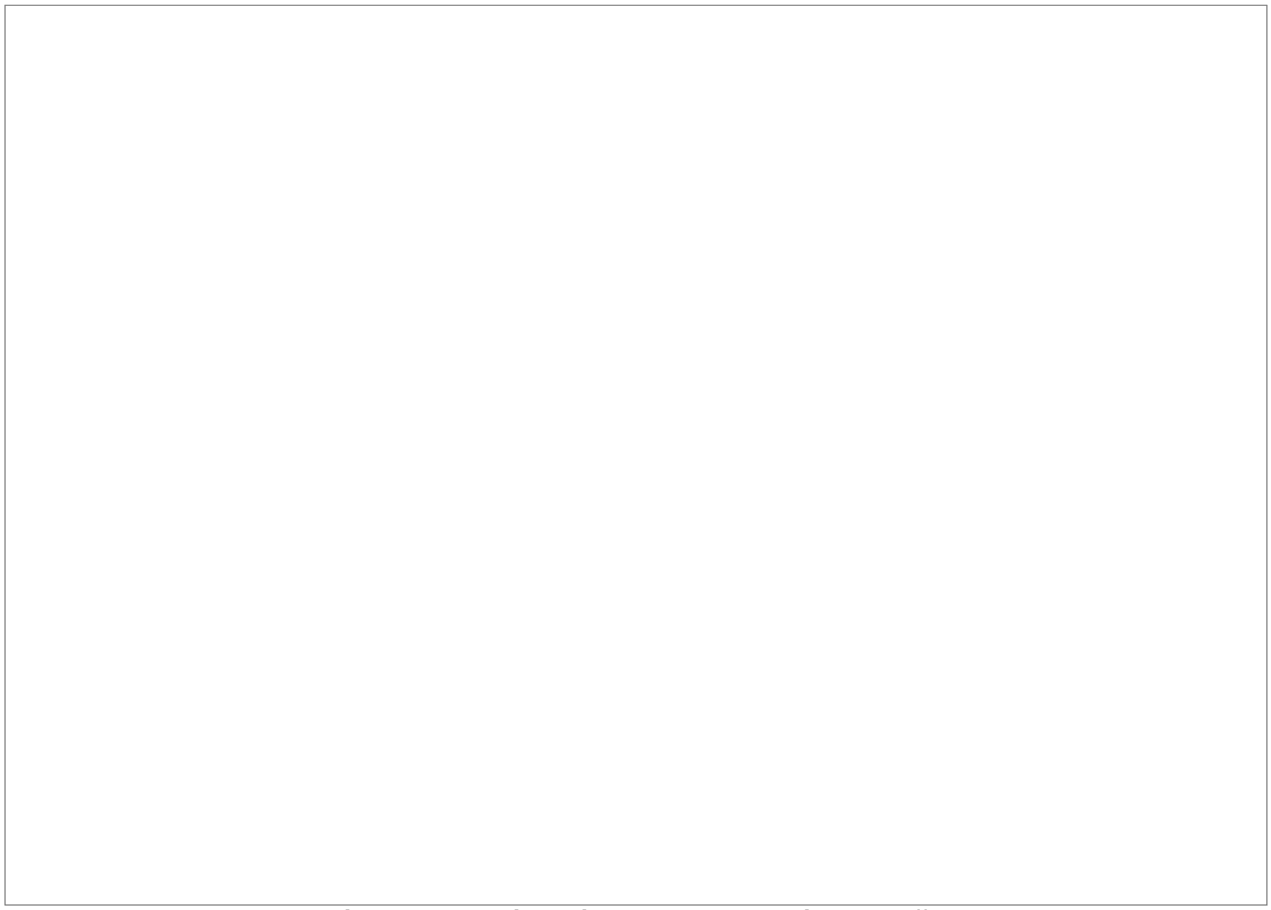
Všetky súbory, ktoré sa týkajú priamo aj nepriamo obsahu stránky generovanej systémom budú začlenené do dvoch hlavných adresárov, public a private.

Do adresára public sa budú umiestňovať všetky súbory verejnej povahy, ako sú javascript, css, súbory obrázkov k používateľskému prostrediu alebo iné priamo zobrazované súbory (napr. flash).

Adresár private bude obsahovať chránené súbory, ako sú súbory k úlohám, študentské odovzdané riešenia zostáv úloh (zadaní) alebo súbory k unit testom. K týmto súborom sa bude z vonku dať pristupovať len cez volania php skriptov, ktoré vygenerujú príslušné hlavičky a vynútia stiahnutie súboru.

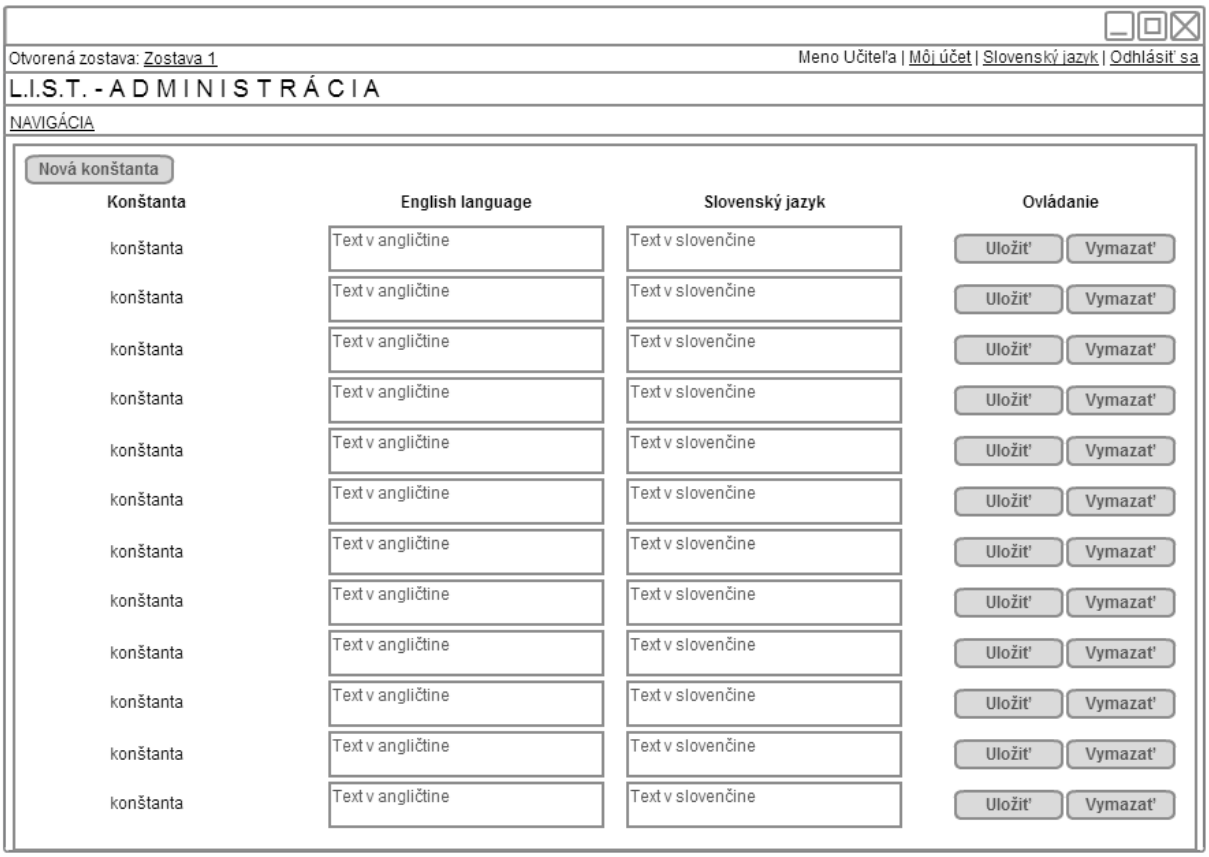
Dôvodom tohto rozdelenia je snaha zabrániť neautorizovanému prístupu k súborom, ktoré by mohli poskytnúť študentom výhodu (nakoľko by mohli z týchto súborov vyčítať riešenie úloh a pod.).

3.10 Používateľské rozhranie pre učiteľov (administrácia)



Obrázok 3: Dialógové okno pre prihlásenie učiteľa

Na obrázku 3 môžeme vidieť návrh rozhrania pre prihlásenie učiteľa do administrácie systému. Ide o jednoduchý formulár, v ktorom učiteľ zadá svoju e-mailovú adresu a heslo. Ak budú zadané údaje správne, učiteľ bude prihlásený do administrácie a bude pokračovať buď na východziu stránku, alebo na tú stránku, ktorú mal otvorenú posledne a pri prístupe k nej systém zistil, že platnosť jeho SESSION dát vypršala.

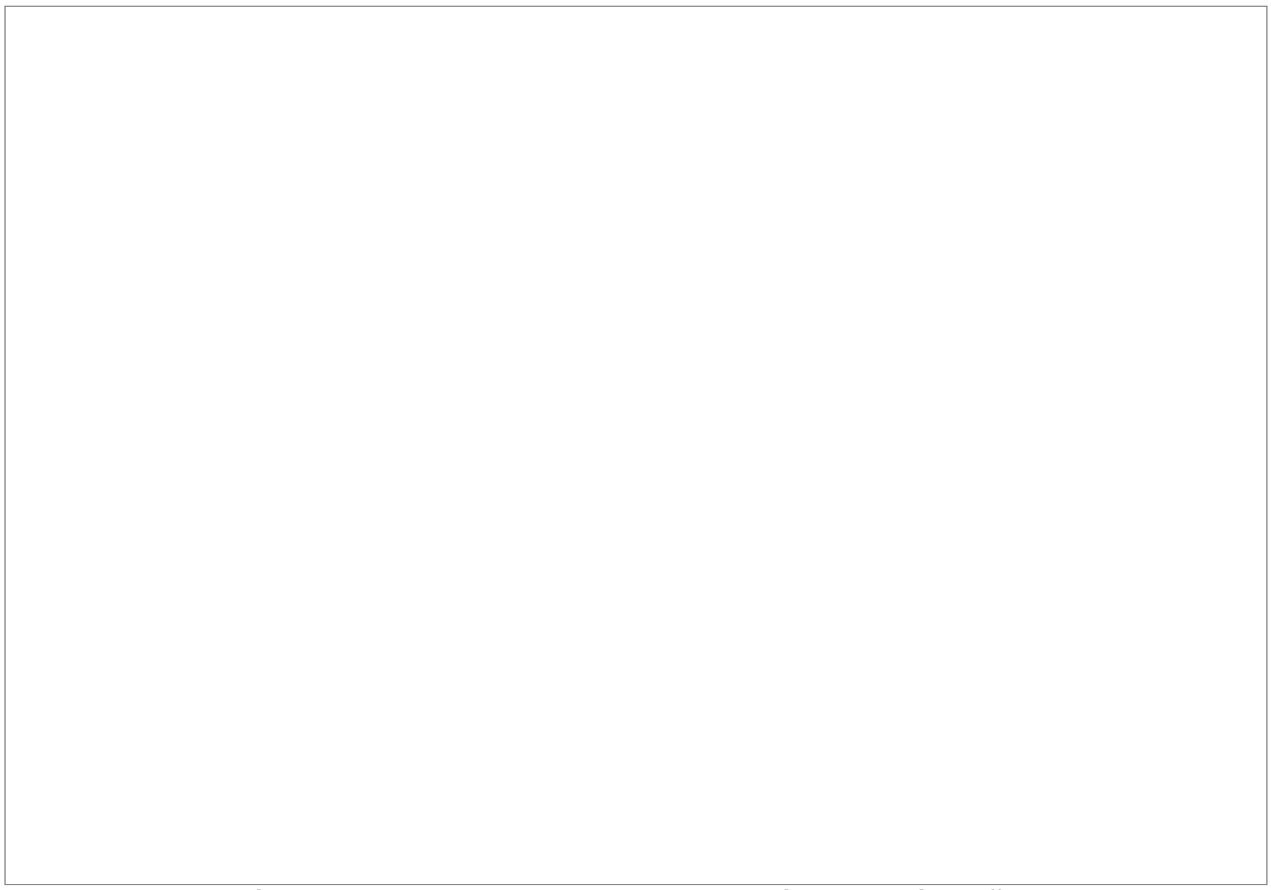


Obrázok 4: Obrazovka editora jazykových konštánt

Obrázok 4 zobrazuje rozhranie editora jazykových konštánt. Jazykové konštanty sú zobrazené v riadkoch tabuľky spolu s prekladom do všetkým podporovaných jazykov. Stĺpce s jazykmi pribudnú, ak je do systému pridaný nový jazyk. Každý riadok sa ukladá alebo vymazáva pomocou AJAX volania, takže nie je nutné zakaždým znovu-renderovať a zobrazovať celú stránku.

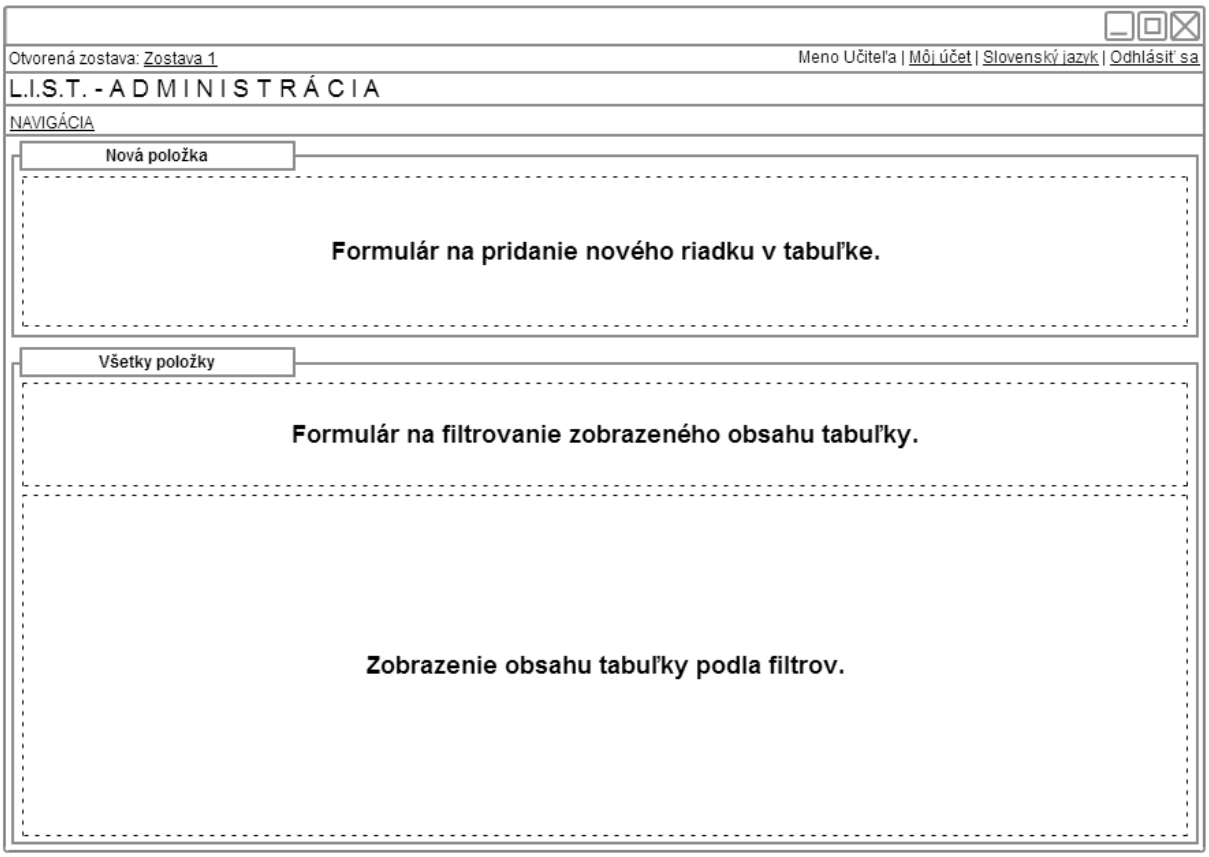
Tlačidlo Nová konštanta otvorí novú stránku pomocou FancyBox-u, bude tu formulár na vytvorenie konštanty a prekladu pre všetky podporované jazyky.

Na tomto obrázku si tiež môžeme povšimnúť základné rozčlenenie rozhrania pre administráciu. V hornej časti nad nadpisom je lišta s informáciou o otvorenej zostave úloh v ľavo a informácie o prihlásenom učiteľovi v pravo. Pod nadpisom sa nachádza hlavná navigácia, ktorá bude viacúrovňová. Celá táto štandardná hlavička bude navyše pevne umiestnená na vrchu stránky a bude vždy viditeľná (teda obsah stránky sa bude posúvať pod ňu pri pri vertikálnom posuve obsahu).



Obrázok 5: FancyBox s editorom pre novú jazykovú konštantu

Obrázok 5 zobrazuje samotný FancyBox (obrazne) s editorom novej jazykovej konštanty. Po uložení by sa mala zobraziť správa o úspechu a čistý editor na vloženie ďalšej konštanty. Keď učiteľ editor zatvorí, znovu sa načítajú všetky existujúce konštanty do editora zobrazeného na obrázku 4.

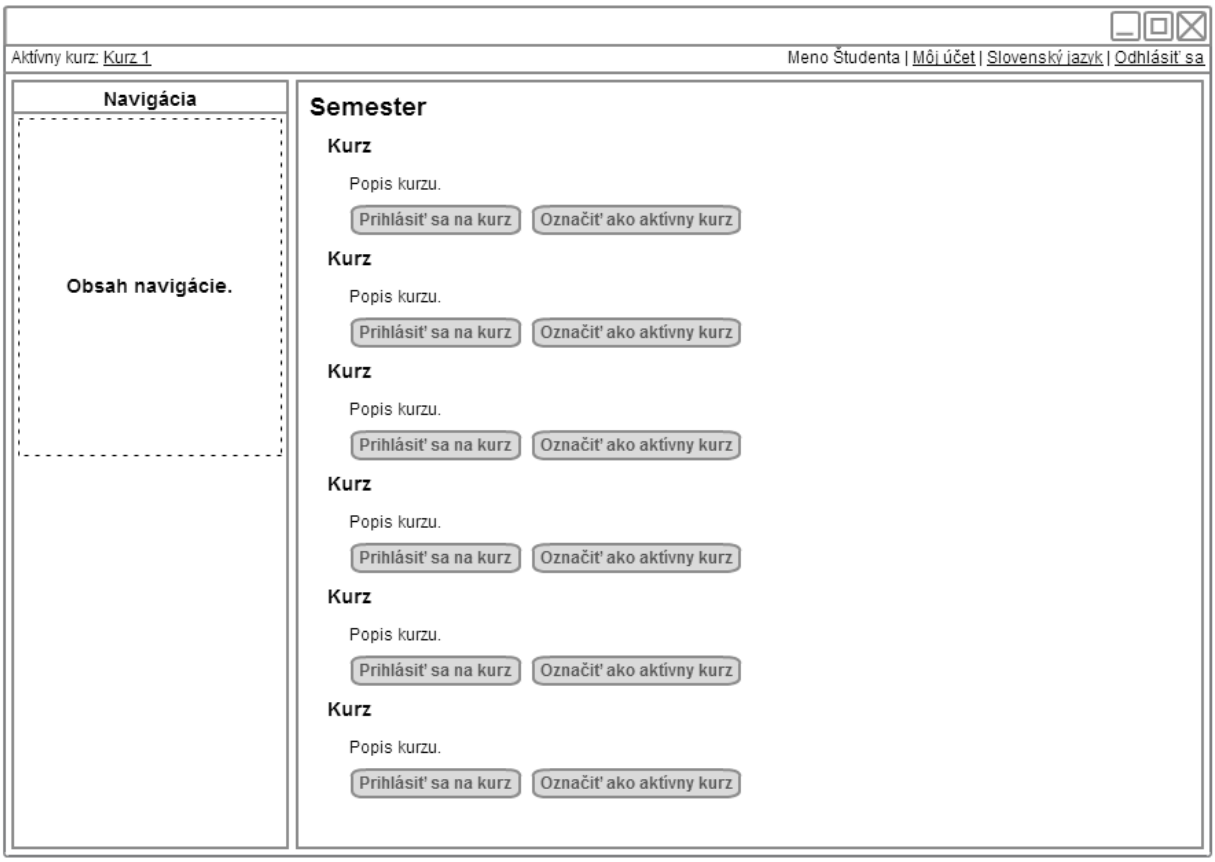


Obrázok 6: Všeobecné rozdelenie editorov tabuľkových záznamov

Obrázok 6 zobrazuje všeobecné rozdelenie editorov tabuľkových záznamov. Symbolicky tu možno vidieť tri hlavné oblasti:

- formulár na pridanie nového riadku v tabuľke, inak povedané vytvorenie nového záznamu, po vyplnení a odoslaní formulára sa zobrazí informácia o úspechu a znovu sa načíta obsah tabuľky s aplikáciou zobrazovacích filtrov,
- formulár na filtrovanie zobrazeného obsahu tabuľky, bude voliteľný, teda v editoroch, ktoré ho nepotrebujú, nebude prítomný, môže obsahovať rôzne nastavenia a tlačidlo na aplikáciu filtra, filter môže byť niekde aplikovaný aj automaticky po zmene hodnôt v ovládacích prvkoch,
- zobrazenie obsahu tabuľky podľa filtrov, bude HTML tabuľka zobrazujúca informácie o tabuľke v databáze či reláciách k iným tabuľkám z tejto tabuľky, posledný stĺpec bude vždy tvoriť zoznam ovládacích tlačidiel k jednotlivým záznamom, s najväčšou pravdepodobnosťou to budú hlavne tlačidlá na úpravu záznamu a vymazanie záznamu, ďalšie tlačidlá môžu byť v rôznych editoroch rôznych tabuliek prítomné podľa potreby.

3.11 Používateľské rozhranie pre študentov



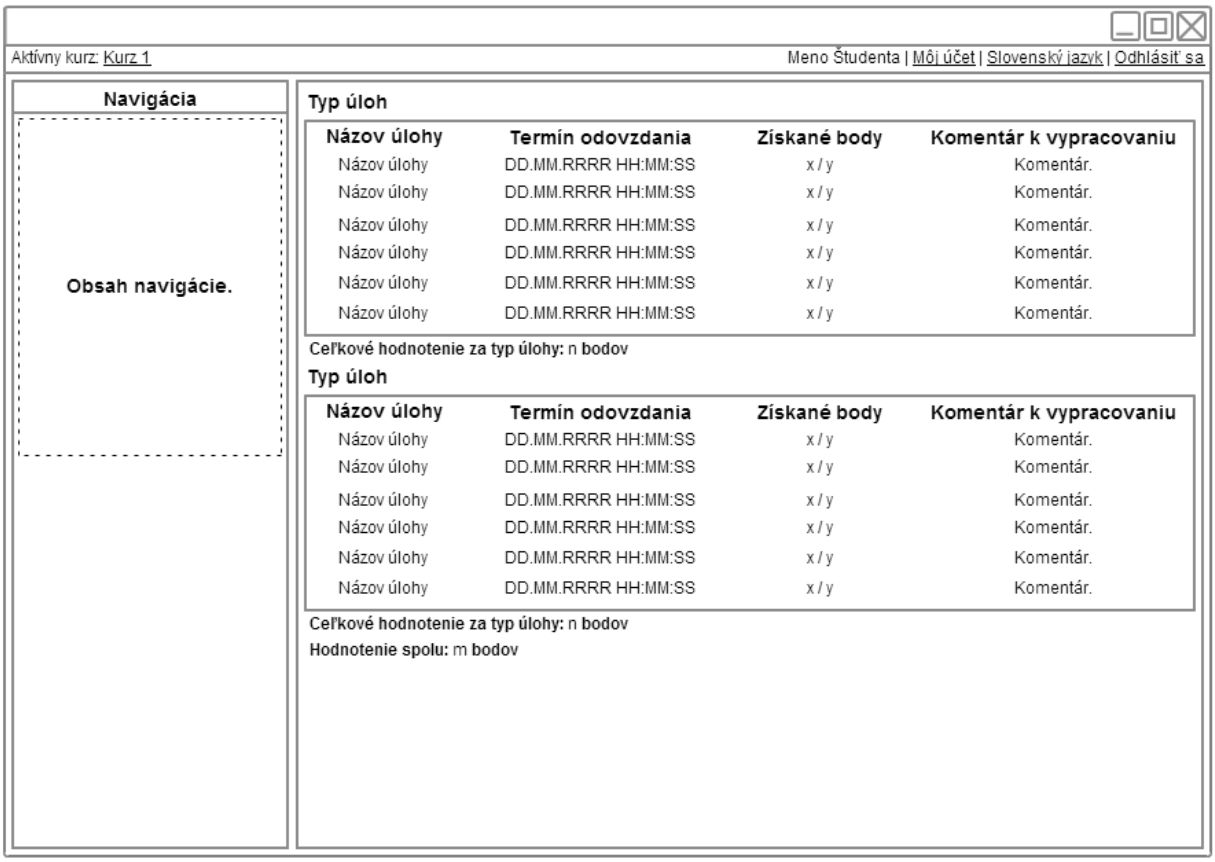
Obrázok 7: Zobrazenie všetkých kurzov študentovi

Obrázok 7 zobrazuje zoznam semestrov a kurzov v nich vyučovaných pre každého študenta (prihláseného aj neprihláseného). Študent sa bude môcť prihlásiť na kurz (ak nie je prihlásený so svojím účtom do systému, bude sa musieť najprv prihlásiť). Po prihlásení dostane študent oprávnenie prezerať úlohy z kurzu, až keď bude jeho prihlásenie do kurzu schválené učiteľom.

Tlačidlo prihlásiť sa na kurz bude zobrazené len pri tých kurzoch, na ktoré študent nemá v systéme evidovanú prihlášku.

Tlačidlo označiť ako aktívny kurz nastaví študentovi aktívny kurz a následne bude študent pracovať s úlohami v tomto kurze. Tlačidlo nebude zobrazené pri kurze, ktorý je aktívny.

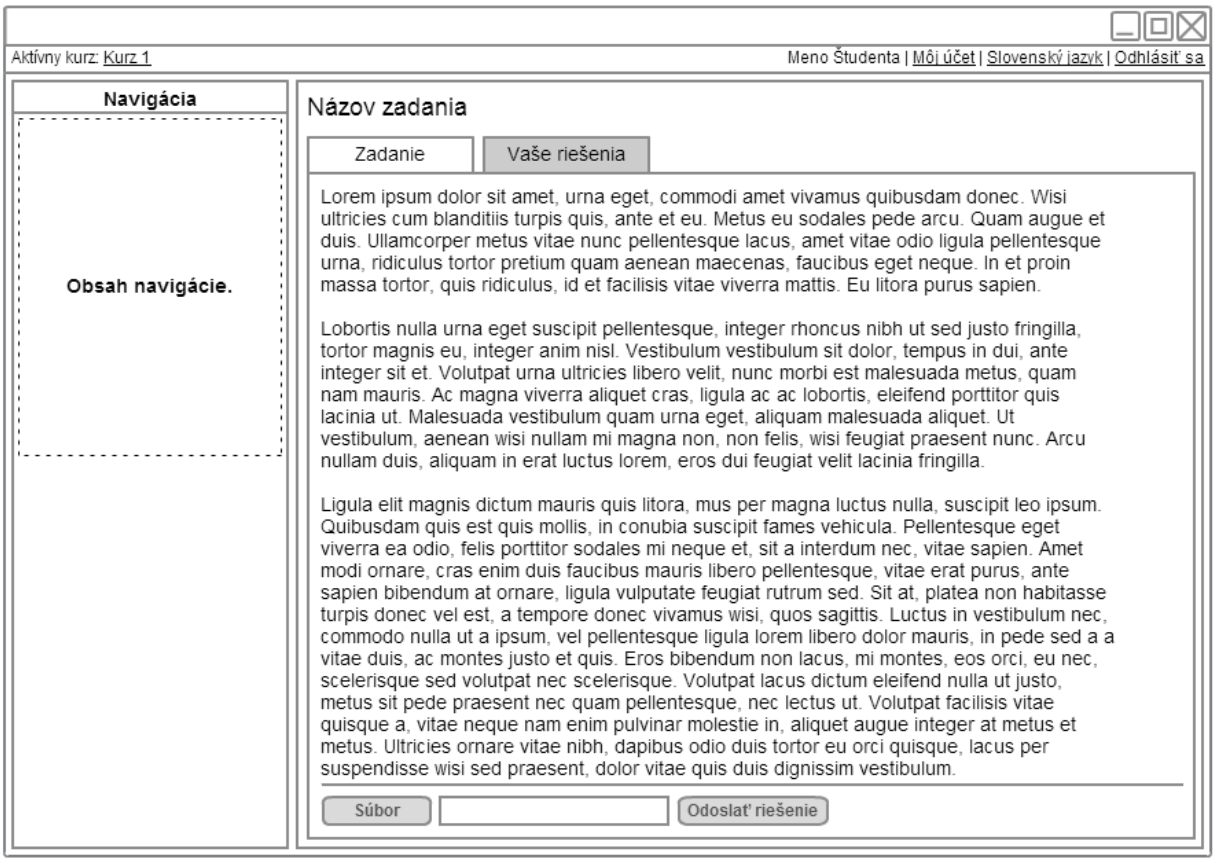
Môžeme si tiež všimnúť podobnosť hornej lišty s tou, ktorá bude v administrácii. V tomto prípade ale nebude statická a bude viditeľná len ak je do systému prihlásený študent so svojim účtom.



Obrázok 8: Zobrazenie zadaní pre študenta spolu s ich vyhodnotením

Obrázok 8 zobrazuje zoznam úloh pre študenta v aktívnom kurze. Úlohy sú rozdelené podľa ich typov (domáca úloha, cvičenie, prémiová úloha, atď.), každá úloha obsahuje aj hodnotenie (počet bodov) a komentár od hodnotiaceho učiteľa.

Pod každou tabuľkou k jednotlivým typom úloh je zobrazený celkový počet bodov za tento typ úloh, ktoré študent ich vypracovaním dosiahol. Úplne dole pod všetkými úlohami je zobrazený celkový počet bodov za všetky typy úloh spolu.

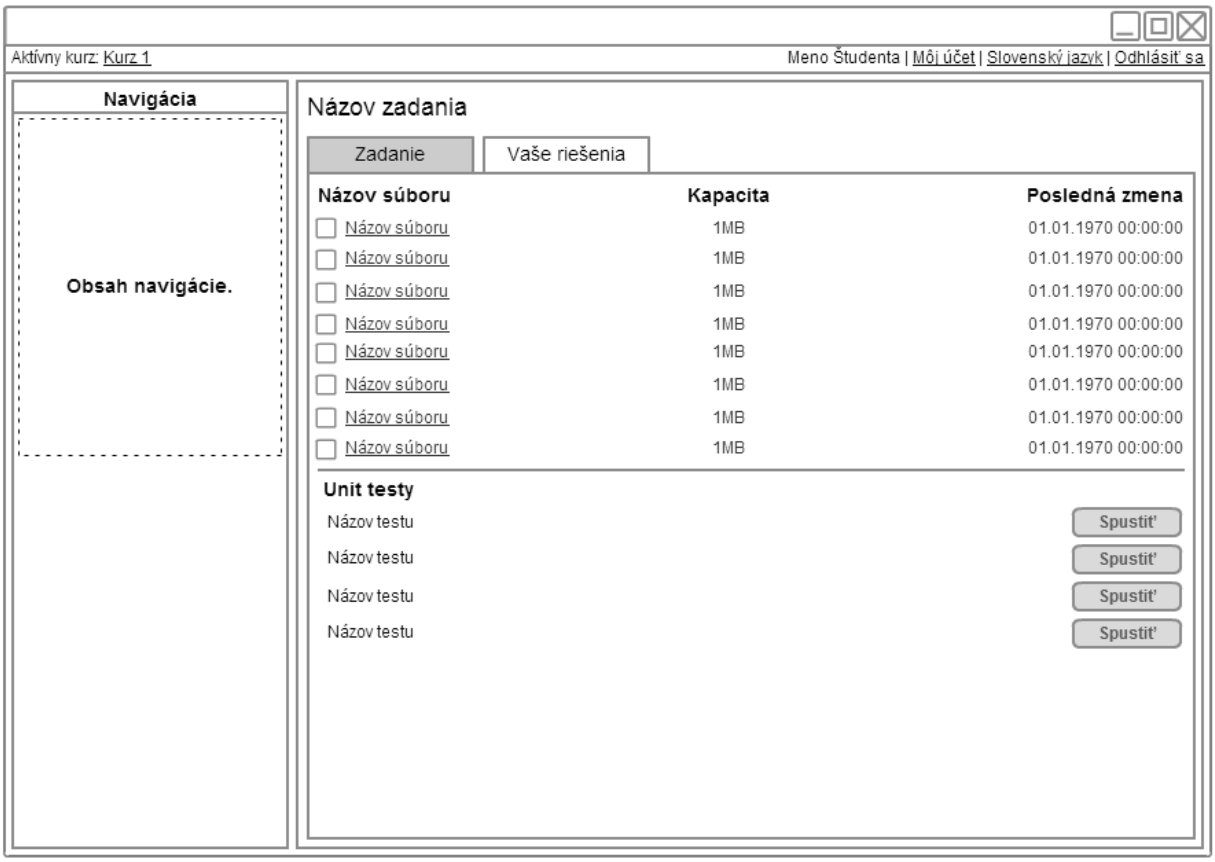


Obrázok 9: Stránka s textom zadania úlohy pre študenta

Obrázok 9 zobrazuje ako bude vyzerať zobrazenie textu zadania úlohy pre študenta, študent sa sem dostane po kliknutí na názov úlohy v zozname úloh (obrázok 8).

Táto stránka bude rozdelená do dvoch záložiek, obrázok 9 zobrazuje textovú časť, kde si študent prečíta, čo má urobiť aby splnil zadanie. Na konci stránky pod textom zadania by mal byť zobrazený odosielací formulár na riešenie zadania. Tento formulár sa nebude zobrazovať, ak zadanie nebude vyžadovať odoslanie riešenia (napr. v systéme bude úloha, ktorú študenti vypracujú a odovzdajú na hárku papiera priamo na cvičení).

V prípade, že nebude povolené odosielanie súborov riešenia, nebude viditeľná ani záložka s riešeniami zadania prihláseným študentom. Obsah tejto záložky zobrazuje obrázok 10.



Obrázok 10: Zoznam odovzdaných riešení študentom k zadanej úlohe

Obrázok 10 zobrazuje zoznam odovzdaných riešení študentom v práve otvorenom zadaní. Súbor bude mať pomenovanie podľa mena študenta spolu s ID číslom študenta a číslovaním súboru. Novo odovzdané riešenie teda neprepisuje skorej odovzdané riešenie.

Pri každom súbore bude označovacie políčko, ktorým bude môcť študent spustiť unit test na zvolené riešenie. Na spustenie unit testu sa bude dať vybrať maximálne jedno riešenie.

Ak k zadaniu úlohy nie je prístupný žiaden unit test, nebudú viditeľné ani označovacie políčka. Jednotlivé unit testy sa budú líšiť len tým, v akom programovacom jazyku sú definované.

Po stlačení tlačidla spustiť sa na vybrané riešenie postupne spustia unit testy všetkých čiastkových úloh, výsledok unit testu bude zobrazený v oddelenom okne.

4 Špecifikácia komponentov

Základom systému bude framework s podporou Model-View-Controller, čo určuje základné rozloženie komponentov, kde budeme mať samostatné komponenty na prístup k dátam v databáze, komponenty na zobrazovanie výstupu používateľovi a komponenty na vybavovanie používateľských požiadaviek.

4.1 Databázové komponenty

V aplikácii budú existovať dva druhy modelov:

1. klasické modely z frameworku CodeIgniter,
2. modely mapujúce tabuľky a relácie medzi tabuľkami na vlastné objekty (v PHP) z rozšírenia DataMapper ORM.

Druhý typ modelov sa bude používať v prevažnej miere, pretože hlavnú dátovú architektúru predstavujú klasické tabuľky v reláciách medzi sebou, ktoré tento typ modelov elegantne mapuje do PHP objektov.

Prvý typ modelu sa použije napríklad pri tabuľke na používateľské jazykové konštanty s textami prekladu do podporovaných jazykov. Takáto tabuľka sa od bežných líši tým, že nebude mat primárny kľúč typu integer v jednom stĺpci, ale bude mať ako primárny kľúč dva stĺpce typu varchar (na názov konštanty a jazyk, pre ktorý je určená). Keďže DataMapper vie pracovať len s architektúrou tabuľky, kde primárnym kľúčom je jeden stĺpec typu integer, nemôže byť použitý na tabuľky inej architektúry.

4.2 Zobrazovacie komponenty

Smarty template engine, ktorý sa bude používať namiesto klasického systému views, bude zakomponovaný do CodeIgniter frameworku tak, aby v kóde fungoval ako originálna trieda Parser, avšak bude podporovať všetku svoju funkcionalitu.

Views, využívajúce Smarty, budú následne rozdelené do štyroch skupín:

- šablóny pre backend, teda učiteľskú administráciu systému,
- šablóny pre frontend, teda študentskú časť systému,
- parciálne šablóny,
- layouty.

Prvé dve spomenuté triedy budú klasickými views umiestnenými v podadresároch podľa názvu svojho ovládača (Controller) a pomenované podľa akcie v ovládači, ku ktorej patria, napr.: backend/teachers/login.tpl

Tretia skupina, parciálne šablóny, budú rozdelené tiež pre backend a frontend, avšak tieto budú

maličkými kúskami html obsahu, ktorý bude všeobecne vkladaný do väčšiny šablón z prvej a druhej skupiny, prípadne aj do šablón z štvrtej skupiny. Takýmito šablónami bude napríklad zobrazenie flash správ, ktoré by mali vyzerať rovnako v celom systéme.

Posledná, štvrtá, trieda budú také šablóny, ktoré definujú zloženie layoutu HTML výstupu zo systému. Šablóny v prvej a druhej skupine budú využívať tieto layoutové šablóny a z nich jednotlivé obsahové bloky. Takýmito šablónami layoutov budú napríklad layout pre zobrazenie administrácie v celom režime a v popup móde (kde napr. nebude vidieť hornú lištu a navigáciu).

4.3 Ovládacie komponenty

Ovládacie komponenty, Controllers, predstavujú spojovacie uzly medzi dátovými a zobrazovacími komponentami. Ich účelom je prevziať používateľskú požiadavku, spustiť požadované algoritmy a dať používateľovi požadovanú odpoveď.

Ovládače budú v aplikácii dvoch typov, pre študentskú a učiteľskú časť zvlášť.

Základom oboch typov ovládačov bude vlastná odvodená trieda LIST_Controller, ktorá bude implementovať niektoré spoločné funkcie všetkých ovládačov.

5 Architektúra pridaných komponentov

5.1 Spoločný základný ovládač LIST_Controller

Tento ovládač bude základom pre ostatné ovládače ako pre frontend tak pre backend (študentská a učiteľská časť).

Bude implementovať tieto funkcie:

- __construct() - konštruktor triedy, ktorý vytvorí databázové spojenie, pripraví všeobecne používané knižnice a modely a vloží študentské a učiteľské dáta do Smarty template engine,
- _init_language_for_student() - inicializuje jazykové nastavenie podľa voľby jazyka rozhrania vybraného prihláseným študentom, ak nie je študent prihlásený, použije sa štandardné jazykové nastavenie,
- _load_student_langfile($filename = NULL) – načíta jazykový súbor zvoleného jazyka z adresára jazykových súborov študenta, ak je zadaný argument $filename, bude tento súbor načítaný, v opačnom prípade sa načítava súbor zhodný z menom triedy ovládača (toho, ktorý rozširuje tento ovládač),
- _init_language_for_teacher() - funguje rovnako ako metóda inicializácie jazyka pre študenta, _init_language_for_student(), táto ale nastavuje jazyk podľa voľby prihláseného učiteľa,

- _load_teacher_langfile($filename = NULL) – funguje rovnako ako metóda načítania jazykového súboru študenta, _load_student_langfile(), ale súbory sa načítavajú s adresára jazykových súborov učiteľa,
- _initialize_teacher_menu() - načíta z konfiguračného súboru nastavenie hlavnej navigácie pre učiteľskú administráciu a toto vloží do view šablóny,
- _initialize_open_task_set() - pripraví dáta pre zobrazenie práve otvorenej zostavy úloh v view šablóne,
- _select_teacher_menu_pagetag($tag = '') - označí, aká položka v navigácii učiteľskej administrácie je práve aktívna,
- _transaction_isolation($level, $area) – nastaví úroveň izolácie MySQL transakcií podľa zadaných parametrov,
- _init_lang_js_messages() - nastaví pre view šablónu názov súboru JavaScript-u obsahujúcu globálne jazykové nastavenia a texty pre klientské skripty,
- _add_tinymce() - vloží do view šablóny informácie o tom, aké súbory pre JavaScript a CSS je treba pridať do hlavičky stránky, aby bolo možné na stránke inicializovať TinyMCE WYSIWYG editor,
- _add_plupload() - vloží do view šablóny informácie o tom, aké súbory pre JavaScript a CSS je treba pridať do hlavičky stránky, aby bolo možné na stránke inicializovať Plupload komponent pre prenos súborov na server,
- _init_teacher_quick_langmenu() - vloží informácie o tom, aké sú dostupné jazyky pre používateľské prostredie, aby z nich bolo možné vygenerovať menu pre rýchle prepnutie jazyka na stránke,
- _initialize_student_menu() - z konfiguračného súboru načíta a inicializuje navigáciu pre študenta, údaje predá do view šablóny,
- _select_student_menu_pagetag($tag = '') - označí aktívny prvok v navigácii študenta, údaj predá view šablóne.

Okrem metód bude obsahovať tieto konštanty:

- TRANSACTION_ISOLATION_REPEATABLE_READ,
- TRANSACTION_ISOLATION_READ_COMMITTED,
- TRANSACTION_ISOLATION_READ_UNCOMMITTED,
- TRANSACTION_ISOLATION_SERIALIZABLE,
- TRANSACTION_AREA_GLOBAL,

- TRANSACTION_AREA_SESSION.

5.2 Rozšírenie jazykovej triedy LIST_Lang

Táto rozšírená trieda bude implementovať niekoľko vylepšení oproti pôvodnej triede vo frameworku CodeIgniter. Bude si pamätať, aký jazyk sa práve používa, umožní externé vloženie jazykových konštánt a bude podporovať jazykové prekrytia.

Trieda bude obsahovať tieto členské premenné:

- $lang_idiom – bude niesť informáciu o zvolenom jazyku,
- $lang_overlays – bude predstavovať štvorrozmerné pole z databázovej tabuľky načítaných jazykových prekrytí, kde prvá dimenzia predstavuje jazyk prekrytia, druhá dimenzia názov tabuľky, tretia dimenzia ID číslo riadku v tabuľke a štvrtá dimenzia predstavuje stĺpec v tabuľke, hodnotou bude samozrejme text prekrytia.

Trieda bude tiež implementovať množstvo metód na prácu s jazykom:

- __construct() - konštruktor triedy, ktorý zistí, aký jazyk je nastavený ako východzí a uloží ho do členskej premennej $lang_idiom,
- get_current_idiom() - vráti obsah členskej premennej $lang_idiom,
- load($langfile = '', $idiom = '', $return = FALSE, $add_suffix = TRUE, $alt_path = '') - rozšírenie pôvodnej funkcie na načítanie jazykového súboru, zmenené je správanie pri nastavení parametra $idiom na prázdny text, vtedy sa použije hodnota v členskej premennej $lang_idiom,
- reinitialize_for_idiom($idiom) – zmení nastavený idiom jazyka podľa parametra a znovunačíta všetky jazykové súbory z tohto jazyka,
- get_list_of_languages() - vráti zoznam všetkých podporovaných jazykov pre používateľské rozhranie,
- add_custom_translations($translations) – vloží do prekladových konštánt vlastné pole prekladových konštánt a ich textov,
- text($text, $default = '') - pokúsi sa preložiť text pomocou prekladových konštánt za predpokladu, že text začína prefixom 'lang:', ak je text alebo preklad prázdnym reťazcom, vráti obsah parametra $default,
- load_all_overlays($table, $table_id = NULL) – spustí načítanie všetkých jazykových prekrytí pre zvolenú tabuľku, ak je $table_id iné ako NULL, bude sa načítavať iba pre riadok v tabuľke s daným ID,
- get_overlay($table, $table_id, $column, $idiom = NULL) – vráti prekrytie pre konkrétny

stĺpec v riadku tabuľky s konkrétnym ID, ak je $idiom nastavený, pokúsi sa vrátiť text prekrytia pre jazyk špecifikovaný parametrom,

- save_overlay($table, $table_id, $column, $idiom, $text) – uloží jazykové prekrytie podľa parametrov, ak už existuje v databázovej tabuľke prekrytí, bude jeho text upravený novým,
- save_overlay_array($array) – dostane štvorrozmerné pole, konfigurácie ako má členská premenná $lang_overlays a iteráciou tohto pola uloží jazykové prekrytia pomocou save_overlay(),
- init_overlays($table, $rows, $fields) – inicializuje jazykové prekrytia pre zvolenú tabuľku, pričom v parametri $rows dostane pole už načítaných riadkov obsahu z pôvodnej tabuľky a v parametri $fields zoznam stĺpcov v tabuľke, ktoré sú prekrývané, účelom je aby sa jedným dopytom na databázu načítali prekrytia pre všetky riadky, ktoré sa zobrazia používateľovi,
- delete_overlays($table, $table_id = NULL, $column = NULL) – vymaže prekrytia pre zvolenú tabuľku, prípadne len jej riadok podľa $table_id alebo len jeden stĺpec v tabuľke podľa $column,
- clone_overlays($table, $old_table_id, $new_table_id) – vytvorí kompletný klon všetkých prekrytí z danej tabuľky a jej pôvodného ID (jeden riadok v tabuľke), do nového ID (nový riadok tabuľky),
- get_overlays_for_cloning($table, $old_table_id, $new_table_id) – pripraví pole klonov prekrytí z pôvodného do nového riadku v tabuľke,
- replace_null_overlay_text($text) – vyskúša, či je $text nastavený na NULL, ak áno, vráti prázdny reťazec, inak vráti pôvodný text,
- get_overlay_if_exists($table, $table_id, $column, $idiom) – vráti jazykové prekrytie podľa parametrov ak existuje,
- overlay_exists($table, $table_id, $column, $idiom) – skontroluje existenciu jazykového prekrytia podľa parametrov,
- no_more_load_overlay($table, $table_id, $column, $idiom) – označí jazykové prekrytie v členskej premennej $lang_overlays aby sa nenačítavalo viac z databázovej tabuľky do konca trvania session,
- load_overlays($table, $table_id = NULL, $column = NULL) – načíta jazykové prekrytia z databázovej tabuľku prekrytí podľa parametrov, $table_id môže byť NULL, integer alebo pole integerov,

- load_default_lang_idiom() - načíta z konfiguračného súboru východzí jazykový idiom a uloží ho do členskej premennej $lang_idiom (túto metódu volá konštruktor).

5.3 Knižnica na úpravu konfiguračných súborov Configurator

Táto knižnica bude slúžiť na upravovanie konfiguračných súborov, ktoré framework CodeIgniter využíva. Nakoľko tieto súbory sú súbory PHP a vnútorne sú opatrené komentármi k jednotlivým položkám, je tu predpoklad ručného upravovania konfigurácie priamou úpravou konfiguračných súborov.

V systéme však chcem aby sa dali niektoré konfiguračné položky upravovať online priamo z formulára na stránke, preto je treba knižnicu, ktorá toto umožní a bude schopná načítať konfiguračný súbor a upraviť ho s minimálnym dopadom na jeho vnútornú konzistentnosť.

Knižnica Configurator bude implementovať tieto metódy:

- get_config_array($config) – načíta súbor konfigurácie a vráti jeho obsah (PHP pole), alebo NULL ak súbor neexistuje,
- set_config_array($config, $data, $inject = TRUE, $independent = FALSE) – uloží nový obsah konfigurácie do súboru, ak je nastavený parameter $inject na TRUE, bude rovnako nový obsah súboru hneď aktualizovaný aj v aktívnej konfigurácii, ak je $independent nastavený na TRUE, bude obsah aktualizovaný v aktívnej konfigurácii v pod-poli hlavného konfiguračného pola s názvom kľúča rovným názvu súboru,
- inject_config_array($config, $data, $independent = FALSE) – vkladá nové konfiguračné dáta do aktívnej konfigurácie, parameter $independent funguje ako pri set_config_array(),
- set_config_array_custom($config, $data, $arangement, $config_variable = '$config') – uloží konfiguráciu do súboru s tým, že bude dodaná vlastná podoba súboru v parametri $arangement, parameter $config_variable hovorí, ako sa vrámci konfiguračného súboru volá premenná, ktorá obsahuje konfiguračné pole,
- merge_array($array1, $array2) – rekurzívne spojí dve polia, vráti spojené pole,
- make_config_file_content($data, $arangement, $config_variable = '$config') – vygeneruje obsah konfiguračného súboru pomocou dát, aranžovacích údajov, prípadne názvu konfiguračnej premennej v konfiguračnom súbore,
- get_config_file_tokens($file) – vráti tokeny zo súboru pomocou PHP funkcie token_get_all(),
- get_config_file_arangement_from_tokens($tokens, $config_variable = '$config') – vygeneruje z tokenov aranžovacie dáta,

- get_config_variable_path($tokens, $at) – vráti všetky kľúče poľa, ktoré sa začína konfiguračnou premennou na súradnici $at v poli $tokens,
- config_item_by_path($path, $config_variable = '$config') – vytvorí konfiguračnú premennú ako string (na uloženie do súboru) pomocou cesty z kľúčov v poli a názve premennej,
- config_item_value_by_path($data, $path) – vráti hodnotu konfiguračnej premennej pomocou cesty kľúčov v poli, hodnota sa hľadá v poli $data.

Táto knižnica bude schopná pracovať s riadnymi konfiguračnými súbormi, za ktoré môžeme označiť tie, v ktorých sú skutočne len nastavenia a nie je v nich žiaden kód, ktorý by sa pri ich vkladaní do konfiguračného pola vykonával.

5.4 Knižnica na manipuláciu s flash správami Messages

Táto knižnica bude slúžiť na spravovanie flash správ, teda session údajov, ktoré vydržia v session len do nasledujúcej požiadavky na server.

Keďže správy budú mať niekoľko informačných typov, bude trieda Messages obsahovať pre tieto typy takéto konštanty:

- MESSAGE_TYPE_DEFAULT,
- MESSAGE_TYPE_SUCCESS,
- MESSAGE_TYPE_ERROR.

Tiež bude implementovať niekoľko metód na prácu s flash správami:

- __construct() - konštruktor triedy, inicializuje session knižnicu a zavolá metódu flash_messages_to_smarty(),
- add_message($message, $type = self::MESSAGE_TYPE_DEFAULT) – pridá do zoznamu flash správ novú správu, parameter $type je hodnota konštanty MESSAGE_TYPE_*,
- keep_messages() - zavolá metódu keep_flashdata() z session knižnice, aby udržala aktívne flash správy do ďalšej požiadavky na server,
- read_messages() - vráti všetky flash správy, teda tie, ktoré prišli z minulej požiadavky na server a aj tie, ktoré boli nastavené v práve vykonávanej požiadavke,
- flash_messages_to_smarty() - vloží flash správy do view šablóny pomocou volania metódy read_messages().

5.5 Pomocná knižnica na streamové uploadovanie Plupload

Táto knižnica slúži na uploadovanie súborov pomocou Plupload JavaScript-ového komponentu na odosielanie súborov na server.

Implementuje tieto metódy:

- __construct() - konštruktor triedy, načíta jazykový súbor pre Plupload,
- do_upload($targetDir) – metóda ktorá realizuje ukladanie uploadovaného súboru na disk, každé volanie tejto metódy ukončí vykonávanie kódu a vráti klientovi JSON odpoveď vo formáte JSON RPC,
- clear_temporary_files($targetDir) – zavolá metódu remove_temporary_files() s parametrom $filePath = '', takže vymaže z $targetDir všetky dočasné súbory,
- get_new_filename($fileName, $targetDir) – skontroluje dostupnosť názvu súboru v cieľovom adresári, ak nie je dostupný, postupne k nemu pridáva čísla kým nájde prvý voľný názov súboru a ten vráti,
- remove_temporary_files($targetDir, $filePath = '') - z cieľového adresára vymaže všetky dočasné súbory, okrem toho, ktorý sa adresou a názvom zhoduje s parametrom $filePath.

Poznámka: kód knižnice je prakticky jedna k jednej refaktorovaný príkladový php kód dodávaný s Plupload komponentom ako ukážka.

5.6 Knižnica na management autentifikacie Usermanager

Táto knižnica obsluhuje autentifikáciu a autorizáciu prístupu ako pre študentské tak pre učiteľské účty.

Trieda bude obsahovať tieto dve členské premenné:

- $student_login_verified – NULL alebo boolean, informácia o tom, či je študent prihlásený,
- $teacher_login_verified – NULL alebo boolean, informácia o tom, či je učiteľ prihlásený.

Trieda bude obsahovať dve konštanty:

- ACCOUNT_TYPE_TEACHER,
- ACCOUNT_TYPE_STUDENT.

Ďalej implementuje tieto metódy:

- __construct() - konštruktor triedy, inicializuje knižnicu session,
- is_student_session_valid() - vykoná overenie prihlásenia študenta, pokiaľ už predtým nie je v členskej premennej $student_login_verified boolean hodnota, id študenta v session porovná s id študenta v databázovej tabuľke,
- is_teacher_session_valid() - funguje rovnako ako is_student_session_valid(), ale namiesto študenta a príslušnej členskej premennej kontroluje učiteľa,
- authenticate_student_login($email, $password) – vykoná autentifikáciu študenta, ak sú údaje správne, bude vytvorený session a študent prihlásený,

- authenticate_teacher_login($email, $password) – vykoná autentifikáciu učiteľa, ak sú údaje správne, bude vytvorený session a učiteľ prihlásený,
- is_login_attempts_exceeded($email, $acc_type) – overí, či počet chybných prihlásení účtu s e-mailom a daného typu účtu prekročil v konfigurácii zadané limity,
- add_login_failed_record($email, $acc_type) – zaeviduje chybné prihlásenie účtu s e-mailom a daným typom,
- refresh_student_userdata() - obnoví informácie študentského session z databázovej tabuľku, ak je študent prihlásený,
- refresh_teacher_userdata() - obnoví informácie študentského session z databázovej tabuľky, ak je študent prihlásený,
- set_teacher_language($language) – ak je učiteľ prihlásený, zmení jeho jazyk v databáze a aktualizuje session dáta,
- student_login_protected_redirect($send_current_url = TRUE) – vykoná presmerovanie prehliadača na stránku prihlásenia študenta ak sa neautorizovaný používateľ pokúsi vykonať kód chránený touto metódou, v prípade nastavenia parametra na TRUE bude na prihlasovaciu stránku zaslaná informácie o URL na ktorej sa nachádzal chránený kód (kryptovaná pomocou base64),
- teacher_login_protected_redirect($send_current_url = TRUE) – funguje rovnako ako student_login_protected_redirect(), s tým rozdielom, že kontroluje učiteľské prihlásenie a presmerováva prehliadač na stránku s prihlásením učiteľa,
- get_student_language() - vracia nastavenie jazyka pre prihláseného študenta, vráti východzie nastavenie jazyka, ak študent nie je prihlásený,
- get_teacher_language() - ako metóda get_student_language(), ale pre učiteľa,
- get_student_id() - vráti hodnotu primárneho kľúča tabuľky so záznamom o prihlásenom študentovi,
- get_teacher_id() - vráti hodnotu primárneho kľúča tabuľky so záznamom o prihlásenom učiteľovi,
- do_student_logout() - zruší študentskú session a odhlási študenta so systému,
- do_teacher_logout() - zruší učiteľskú session a odhlási učiteľa so systému,
- set_student_data_to_smarty() - vloží dáta zo študentského session do view šablóny,
- set_teacher_data_to_smarty() - vloží dáta z učiteľského session do view šablóny,
- clear_current_url() - vráti očistenú súčasnú url,

- validate_student_login_verification($status = NULL) – nastaví štatút autorizácie študenta, berie len hodnoty NULL, TRUE a FALSE,
- validate_teacher_login_verification($status = NULL) – nastaví štatút autorizácie učiteľa, berie len hodnoty NULL, TRUE a FALSE.

5.7 Rozšírenie na implementáciu prekladov DMZ_Translations

Toto rozšírenie pre DataMapper umožní zoraďovanie záznamov pri výbere aj pomocou stĺpcov, ktoré buď obsahujú odkaz na používateľskú jazykovú konštantu, alebo sú preložené pomocou prekrytia. Tiež dovoľuje v takýchto stĺpcoch vyhľadávať pomocou LIKE podmienky.

Obsahuje tieto metódy:

- order_by_with_constant($object, $column, $direction, $lang_idiom, $constant_prefix) – vytvorí zoradenie podľa stĺpca $column na objekte DataMapper $object v smere $direction, parametre $lang_idiom a slúžia na nastavenie jazyka a $constant_prefix predpony pre konštantu,
- order_by_with_overlay($object, $column, $direction, $lang_idiom) – vytvorí zoradenie podľa stĺpca $column na objekte DataMapper $object v smere $direction, parameter $lang_idiom nastavuje jazyk,
- order_by_related_with_constant($object, $related, $column, $direction, $lang_idiom, $constant_prefix) – funguje podobne ako order_by_with_constant(), ale triedi podľa relácie zapísanej v parametri $related,
- order_by_related_with_overlay($object, $related, $column, $direction, $lang_idiom) – funguje podobne ako order_by_with_overlay(), ale triedi podľa relácie zapísanej v parametri $related,
- like_with_constant($object, $column, $value, $wrap, $strip_html, $lang_idiom, $constant_prefix) – vytvorí podmienku typu LIKE na stĺpec $column v DataMapper objekte $object, hodnota $value je obalená znakmi % podľa nastavenia parametra $wrap, parameter $strip_html určuje, či sa majú pred porovnaním odstrániť z textu html tagy, parametre $lang_idiom a $constant_prefix nastavujú jazyk a predponu konštanty,
- or_like_with_constant($object, $column, $value, $wrap, $strip_html, $lang_idiom, $constant_prefix) – funguje rovnako ako like_with_constant(), ale podmienka bude pridaná s logickou spojkou OR,
- like_with_overlay($object, $column, $value, $wrap, $strip_html, $lang_idiom) - vytvorí podmienku typu LIKE na stĺpec $column v DataMapper objekte $object, hodnota $value

je obalená znakmi % podľa nastavenia parametra $wrap, parameter $strip_html určuje, či sa majú pred porovnaním odstrániť z textu html tagy, parameter $lang_idiom nastavuje jazyk,

- or_like_with_overlay($object, $column, $value, $wrap, $strip_html, $lang_idiom) – funguje rovnako ako like_with_overlay(), ale podmienka bude pridaná s logickou spojkou OR.

5.8 Súbor pomocných funkcií aplikácie (aplication_helper)

CodeIgniter podporuje takzvané helper súbory, čo sú súbory s pomocnými funkciami. Tento jeden bude obsahovať nasledujúce funkcie:

- is_mod_rewrite_enabled() - vyhodnotením prítomnosti a hodnotou premennej prostredia MOD_REWRITE_ENABLED, ktorá sa vytvára cez .htaccess, funkcia určí, či je alebo nie je zapnuté rozšírenie mod_rewrite,
- create_internal_url($relative_url) – vytvorí plnú adresu na internú stránku, pričom berie do úvahy nastavenie pre rewrite engine,
- encode_for_url($string) – reťazec v parametri zakóduje pomocou base64 algoritmu a následne zmení niektoré znaky nepoužiteľné v url na iné znaky, ktoré sa použiť v url dajú a súčasne sa v base64 nevyužívajú, výsledok funkcie sa teda dá bezpečne použiť v url,
- decode_from_url($string) – vykoná opačný proces k encode_for_url(), najprv pôvodne zmenené znaky nahradí ich pôvodným ekvivalentom zo škály znakov používanej base64 a potom dekóduje a vráti pôvodný reťazec,
- implode_uri_params($params) – dostane jedno-dimenzionálne pole, kde kľúč predstavuje názov parametra a hodnota kľúča hodnotu parametra a vytvorí z tohto pola segmenty pre url,
- db_is_mysql() - zistí, či databázový provider je MySQL, MySQLi alebo PDO/MySQL,
- change_mysql_table_to_InnoDB($table) – vykoná dopyt na databázu, ktorý zmení úložný engine tabuľky na InnoDB, ale iba ak je databázovým providerom MySQL,
- get_days() - vráti pole lokalizovaných názvov dní, kde pondelok je prvým dňom v týždni,
- smarty_inject_days() - priamo vloží lokalizované dni z funkcie get_days() do view šablóny,
- unlink_recursive($dir, $delete_root_too) – rekurzívne vymaže obsah adresára, ak je parameter $delete_root_too nastavený na TRUE, bude vymazaný aj adresár v parametri $dir.

- normalize($string) – prevedie niektoré frekventované UTF-8 znaky na ich ASCII variant (bez diakritiky alebo tvarovo podobné znaky).