Katedra Informatiky

Fakulta matematiky, fyziky a informatiky

Univerzita Komenského, Bratislava

Systém pre podporu vyučovania aritmetiky na základných školách

(bakalárska práca)

Autor: Peter Laubert
Školiteľ: doc. RNDr. Daniel Olejár, PhD.

Bratislava 2007

Systém pre podporu vyučovania aritmetiky na základných školách

BAKALÁRSKA PRÁCA

Peter Laubert

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KATEDRA INFORMATIY

9.2.1 Informatika

Doc. RNDr. Daniel Olejár, PhD.

BRATISLAVA 2007

Poďakovanie

Ďakujem vedúcemu bakalárskej práce doc. RNDr. Danielovi Olejárovi PhD. za vedenie, odbornú pomoc a užitočné pripomienky. Ďalej by som rád poďakoval Miroslavovi Mahdoňovi, rodičom a všetkým ostatným čo prispeli k dokončeniu tejto bakalárskej práce.

Čestné prehlásenie

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne, s použitím uvedenej literatúry.

Bratislava, jún 2007

Abstrakt

Obsahom tejto bakalárskej práce je návrh a implementácia systému pre podporu vyučovania aritmetiky na základných školách. Cieľom systému nebolo vytvoriť vhodné podmienky len pre žiaka, ale aj pre učiteľov, ktorý majú vďaka uchovávaniu dát o procese učenia žiakov možnosť sledovať a analyzovať priebeh učenia a vďaka tomu zohľadniť pokrok jednotlivých žiakov a prispôsobiť výučbu ich individuálnym potrebám.

Kľúčové slová: systém pre podporu vyučovania aritmetiky, web technológie, proces tvorby softvéru

Obsah

I. Úvod

Pre matematiku sa používa množstvo adjektív, ale väčšina ľudí by asi povedala, že to je ťažká veda, a len málokto by povedal, že je potrebná. Pritom matematika vznikla na základe praktických potrieb ľudí merať (vzdialenosti, plochy, objemy, váhy, rýchlosti) a obchodovať. Zo svojej praktickej použiteľnosti matematika nestratila ani po stáročiach svojho vývoja. Jej aplikácie možno dnes vidieť prakticky všade. Ako povedal N. I. Lobačevskij: *„Neexistuje ani jedna oblasť matematiky, a to nech je akokoľvek abstraktná, ktorá by sa raz nedala aplikovať na javy reálneho sveta.*" Kto by tomu neveril nech sa pozrie, v koľkých rôznorodých odboroch sa matematika používa. Nájdeme ju v skoro všade - od medicíny cez filozofiu až po umenie.

Keď si uvedomíme význam matematiky pre každodenný život, je zrejmé, že by bolo dobré matematiku ovládať. U väčšiny z nás sa matematická príprava začala na základnej škole. Ako už názov školy napovedá, tu do nás vštepovali základy matematiky. Tie by mali byť pevné, dosť široké a stabilné, aby sme na nich mohli ďalej v budúcnosti stavať. Podobne ako pri dome, keď sú základy kvalitné, môžeme na nich postaviť veľký dom, ktorý sa nezborí pod vlastnou váhou. Ich stavba však stojí čas. Nemôžme hneď vyliať všetok betón, lebo by mohli vniknúť vnútorné trhliny. Navyše po každej etape treba nechať stavbu kúsok postáť, kým sa utrasie a nájde si svoju pozíciu. A obdobne je to s výučbou. Nemôžme žiakom podať všetky poznatky naraz. Musíme ich dávkovať a potom im dať čas, aby si ich osvojili.

Proces učenia riadia v školách pedagógovia. Aby zaistili čo najlepšie výsledky vzdelávania, vyvinuli rôzne techniky a pomôcky. Tieto sa neustále vyvíjajú vďaka novým možnostiam.

V súčasnosti žijeme v informačnom veku a jednou z možností, ktoré nám táto doba poskytuje, sú počítače. Tie si už našli cestu aj do školstva, no ešte stále sa dá vylepšovať použitie. Preto som bol rád, že som si mohol vybrať bakalársku prácu s témou Systém pre podporu vyučovania

aritmetiky na základných školách. Sľubujem si od nej vytvorenie nového prostriedku, ktorý by mohol pomôcť pedagógom vo výučbe takého pekného predmetu akým je práve matematika.

Práca je členená do viacerých kapitol.

V druhej kapitole je teoreticky vysvetlený proces tvorby systému ako aj praktické napĺňanie tejto teórie v rámci vývoja zadaného systému.

V tretej kapitole sú zhrnuté výhody tohto systému v prípade jeho nasadenia do praxe.

Štvrtá kapitola pojednáva o vízii do budúcnosti, poukazuje na možné rozšírenia systému.

Piata kapitola stručne zhrnie výsledky práce.

II. Proces tvorby systému

Každý softvérový systém prechádza určitými etapami, ktoré majú zabezpečiť jeho kvalitu a funkcionalitu. Tieto sú vo väčšine projektov rovnaké, ale líšia sa rozsahom či technikami v nich použitými. Medzi základné rátame štyri:

- Špecifikácia
- Vývoj = návrh + implementácia
- Validácia
- Evolúcia

V nasledujúcom texte uvádzame popis prvých dvoch etáp; t.j. ako prebiehala špecifikácia a vývoj systému, aké techniky boli pri tom použité.

1. Špecifikácia

Špecifikácia systému je aktivita, ktorej cieľom je vytvoriť ucelený popis požiadaviek na systém. Táto fáza predchádza tvorbe každého softvérového systému, či si to tvorcovia, zadávatelia uvedomujú alebo nie. Ale jej vedomé naplnenie však môže urýchliť vývoj softvérového produktu a zabrániť nezrovnalostiam ako pri vývoji, tak aj pri samotnom používaní.

Výstupom špecifikácie je dokument Špecifikácia požiadaviek, ktorý obsahuje popis funkcií produktu, popis údajov, s ktorými produkt pracuje a ostatné vlastnosti produktu respektíve obmedzenia naň. Špecifikácia požiadaviek je záväzná ako pre zadávateľa, tak aj pre riešiteľa

Pre popis požiadaviek je možné si vybrať z viacerých prostriedkov opisu. Pri tomto projekte boli požiadavky zachytené neformálne, prirodzeným jazykom.

Systém má umožniť kontrolovanú výučbu elementárnych aritmetických operácií s celými číslami žiakov prvých ročníkov základných škôl. Systém má mať nasledujúcu funkcionalitu:

1. Interaktívne generuje pomocou generátora náhodných čísel zadania jednoduchých aritmetických operácií (napr. sčítanie, odčítanie, násobenie a delenie) nad celými číslami, spracováva odpovede žiaka a vyhodnocuje ich správnosť.
2. Zaznamenáva čas strávený nad jednotlivými úlohami aj celkový čas potrebný na vyriešenie všetkých úloh.
3. Zaznamenáva výsledky žiaka (úspešnosť a nesprávne odpovede).
4. Umožňuje súborné spracovanie výsledkov učiteľom:
    a. Porovnanie úspešnosti toho istého žiaka pri riešení rozličných úloh.
    b. Porovnanie rôznych žiakov.
    c. Identifikáciu problematických úloh.
5. Umožňuje učiteľovi nastavovať parametre úloh (rozsah čísel, počet úloh, typ operácií), prípadne zadávať špeciálne úlohy (napr. tie, ktoré žiakovi robili problémy).
6. Umožňuje spravovať používateľov (žiakov aj učiteľov; pričom žiakov v systéme spravuje ich učiteľ).
7. Grafické rozhranie, prostredníctvom ktorého systém komunikuje s používateľmi (učiteľom a žiakmi).
8. Perspektívne by systém mal byť rozšíriteľný o ďalšie moduly (napríklad štatistické vyhodnocovanie výsledkov).

2. Vývoj

Etapa vývoja sa rozdeľuje na dve nezávislé fázy – návrh a vývoj. Ako už názvy naznačujú pri návrhu sa vytvorí skôr teoretické pozadie, skica na základe ktorej sa vo fáze vývoja vychádza.

2.1 Návrh

Fáza návrhu určuje ako bude daný softvér realizovaný. Tento proces prechádza od najhrubších po najjemnejšie štruktúry. Ako prvé sa program rozdelí na subsystémy, následne tie na moduly a u tých sa navrhnú triedy respektíve funkcie.

V tejto fáze sa tiež navrhuje používateľské prostredie ako aj štruktúra databázy.

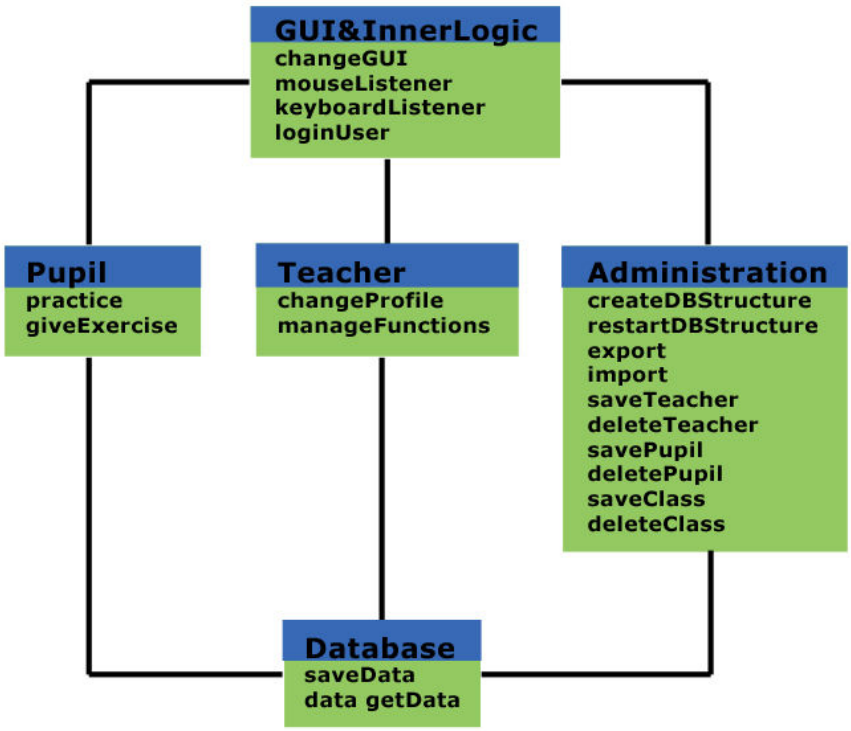
Navrhovanie tohto systému

Pri návrhu tohto konkrétneho systému sme použili modelovací jazyk UML vhodný na modelovanie návrhu, keďže je nezávislý od implementačného jazyk. Táto vlastnosť bola veľmi dôležitá, lebo v tejto fáze tvorby ešte nepadlo rozhodnutie o tom, ktorý programovací jazyk použiť pri implementácii systému.

Keďže pri tomto systéme sa nejedná o rozsiahly projekte, ako je napr. operačný systém, nepridržiavali sme sa úplne postupu, ktorý je uvedený vyššie. Hneď po podrobnom prečítaní špecifikácie sme systém rozdelili do 5 modulov a tým určili ich približnú funkcionalitu:

1. **Žiak (Pupil)** – Tento subsystém má zastrešovať všetku prácu žiaka so systémom, napr. generovanie príkladov a to aj na základe spätnej väzby, vyhodnocovanie.
2. **Učiteľ (Teacher)** - Tento subsystém má zastrešovať všetku prácu učiteľa so systémom, napr. monitorovanie a štatistika výsledkov svojich žiakov, manažment funkcii.

3. **Administrácia (Administration)** – Tento subsystém zastrešuje administráciu systému, napr. pridávanie/odstraňovanie učiteľov, tried a žiakov, export/import databázy.
4. **Databáza (Database)** – Tento subsystém bude spravovať dáta programu ako sú štruktúra užívateľov, výsledky žiakov.
5. **Grafické rozhranie a vnútorná logika (GUI&InnerLogic)** – Tento subsystém bude zastrešovať spracovanie požiadaviek od používateľa, a ich grafické prevedenie.



Obr.1 UML diagram návrhu

Štruktúra systému a vzťahy jednotlivých modulov systému sú uvedené na obrázku 1.

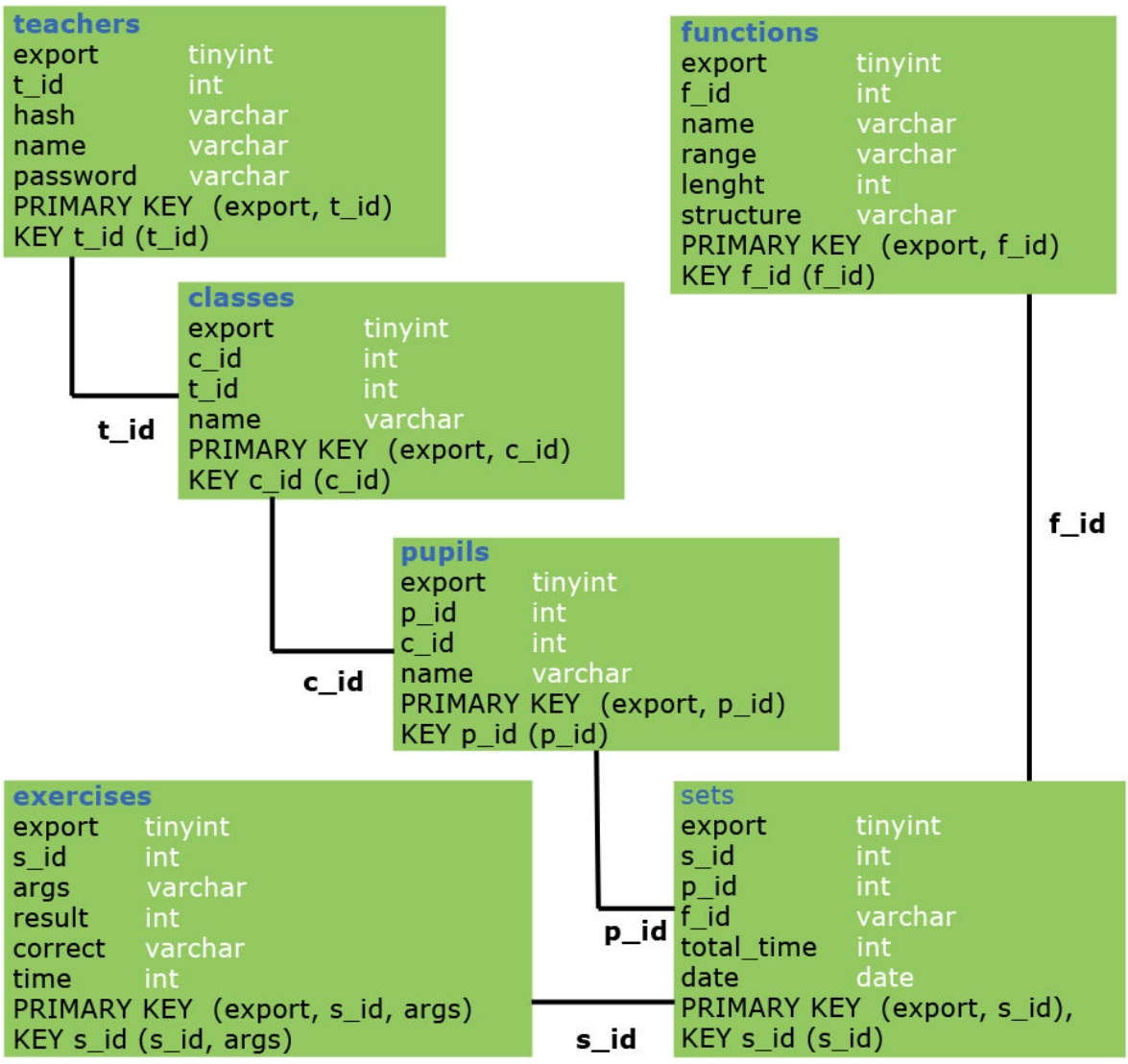
Ďalším krokom návrhu bolo navrhnutie databázy. Systém musí uchovávať nasledujúce informácie o:

- **Učiteľoch (Teachers)** – meno, heslo
- **Triedach (Classes)** – meno, triedny učiteľ
- **Žiakoch (Pupils)** – meno, príslušnosť k triede
- **Funkciách (Functions)** – meno, aritmetický výraz, rozsah, počet príkladov v sade

- **Sadách (Sets)** – príslušnosť k žiakovi, príslušnosť ku funkcii
- **Príkladoch (Exercises)** – príslušnosť k sade, argumenty aritmetického výrazu, výsledok, čas na výpočet

Pre každú z vyššie uvedených entít (učiteľ, trieda, žiak, atď.) bola vytvorená jedna tabuľka. Údaje vzťahujúce sa na tieto entity boli použité ako atribúty tabuliek. Nakoniec tohto kroku sme ešte pridali ďalšie atribúty, ktoré mali buď rozšíriť funkcionalitu, bezpečnosť alebo optimalizovať databázu a to tak že do nej budú uložené informácie, ktoré sa takto vypočítajú len raz, namiesto toho aby sa vypočítavali za každým keď budú vyžadované.

Štruktúra databázy je uvedená na obrázku 2.



Obr.2 Konečná štruktúra databázy

2.2 Implementácia

V tejto fáza vývoja sa vytvára programové dielo. V prvom rade je potrebné vybrať vhodný programovací jazyk, v ktorom sa softvérový systém bude robiť. Toto rozhodnutie závisí od viacerých faktorov:

- Skúsenosti programátorov.
- Dostupnosť know-how.
- Úroveň abstrakcie.
- Knižnice.
- Podporné nástroje.

Po zvolení programovacieho jazyka, v ktorom je vhodné softvér vyvíjať, nastupuje fáza tvorby. Pri tvorbe ako takej je dobré sa pridržiavať niekoľkých pravidiel:

- Štandardizácia pomenovaní konštánt, premenných, tried, funkcií a metód.
- Jednotné, zrakovo prehľadné formátovanie. Toto je hlavne dôležité pri vnoreniach, napríklad pri používaní if-podmienok.
- Komentáre časti kódu. Komentovanie nemá byť doslovný prepis kódu do prirodzeného jazyka, ale malo by byť o úroveň abstrakcie vyššie.
- Vyhnutie sa programátorským trikom. V prípade ich použitia vhodne okomentovať ako a prečo fungujú.

Pre tieto pravidlá neexistujú nejaké univerzálne predpisy a existuje viacero podôb ich prevedenia. Aj cez tento fakt nejde o samoúčelné pravidlá. Ich význam sa napríklad prejavuje pri viac členných týmoch či pri čítaní častí kódu po dlhšom čase.

Ďalšou súčasťou tejto fázy by malo byť dokumentovanie. Ide o dôležitý, aj keď často prehliadaný proces, ktorého úlohou je zdokumentovanie vnútorného dizajnu softvéru pre potreby budúcej údržby a zdokonalenia. Je priam kritické z pohľadu budúcich úprav mať zdokumentované súčasti, ktoré slúžia na vonkajšiu komunikáciu.

Implementovanie systému pre vyučovanie arimetiky

Už pri návrhu sme uvažovali dva programovacie jazyky, v ktorých by bolo vhodné systém implementovať; Java s použitím Derby databázy alebo PHP s MySQL databázou kombinované s iným web technológiami.

Obe možnosti mali viacero spoločných výhod:

- **Multi-platformovosť** – Programy naprogramované v jednom zo spomínaných jazykov sú spustiteľné na viacerých platformách bez potreby dodatočného programovania či modifikácie. Preto sú vhodné do nehomogénneho prostredia počítačového vybavenia škôl.
- **Relačná databáza** – Existencia a možnosť rýchlo implementovať relačnú databázu, pri Jave Derby a pri PHP MySQL, obe pracujúce nad jazykom SQL. Pre použitie relačnej databázy sme sa rozhodli kvôli štruktúre a previazanosti dát, ktoré mali byť ukladané.
- **Dostupnosť know-how** – Pre oba jazyky existuje široká vedomostná základňa obsahujúca dokumentáciu jazyka, manuály, príklady, tutoriály a fóra.
- **Knižnice & podporné nástroje** – Podobne ako pri know-how aj v tejto oblasti je spektrum produktov široké. Napríklad už spomínané relačné databázy patria do tejto skupiny.
- **Skúsenosť** – S oboma technológiami sme mali predošlé skúsenosti, čím sa mi skracoval čas na programovanie a znižovalo sa riziko programátorských chýb.

Pre rozhodnutie však neboli dôležité podobnosti, ale práve rozdiely. Hlavný rozdiel, ktorý nakoniec rozhodol bola ľahkosť akou by sa dal výsledný program modifikovať v prípade sieťového (internetového)-lokálneho použitia. V tomto ohľade podľa nášho názoru dominuje PHP a preto sme si ho vybrali ako implementačný jazyk.

Dizajn

Na účel rozčlenenia stránky sme použili jazyky HTML a CSS. Pomocou CSS sme priradili atribúty HTML tagom body a div, ktoré rozdelili stránku na viacero celkov:

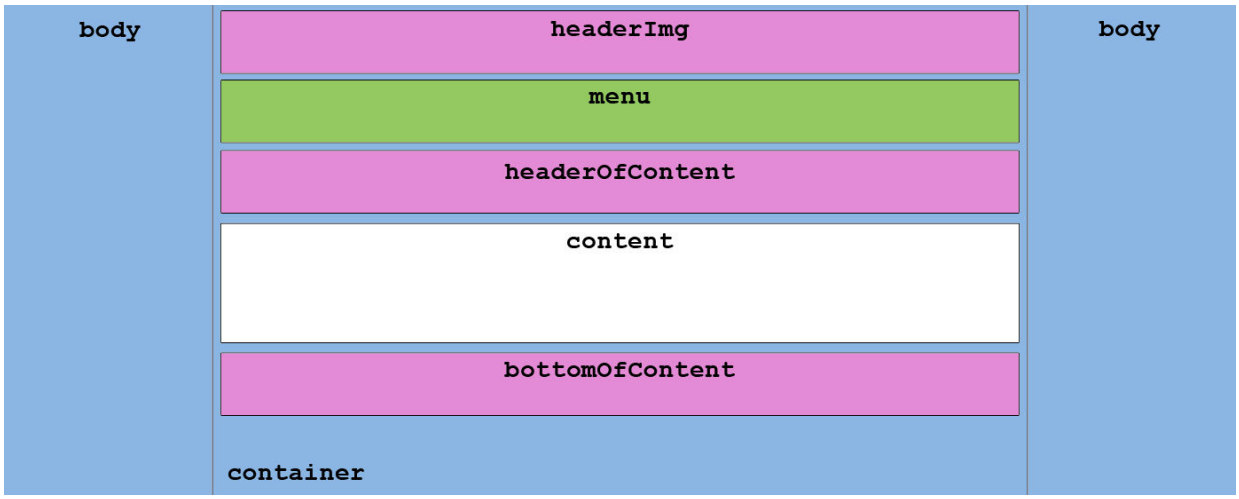
```
<body>
   <div id="container">
         <div><img header" /></div>
         <div id="menu">
         </div>
         <div><img headOfContent /></div>
         <div id="content">

         </div>
         <div><img bottomOfContent /></div>
   </div>
</body>
```

Div container zabezpečuje centrovanie funkčných častí na stred. Div menu obsahuje navigáciu po stránke a div content slúži ako hlavná časť, v ktorej sú vypisované údaje, texty. Tento div má nastavenú minimálnu výšku 400 pixelov, ktorá keď sa prekročí začne sa automaticky plocha rozširovať.

Grafické prvky img boli použité aby navodili dojem oblosti. Museli byť zaobalené do , v opačnom prípade by ich IE nevedel umiestniť do stránky bez vytvárania nechcených medzier.

Koncepcia tejto štruktúry je na obrázku 3 a na obrázku 4 jej reálne prevedenie vykreslené v prehliadači.

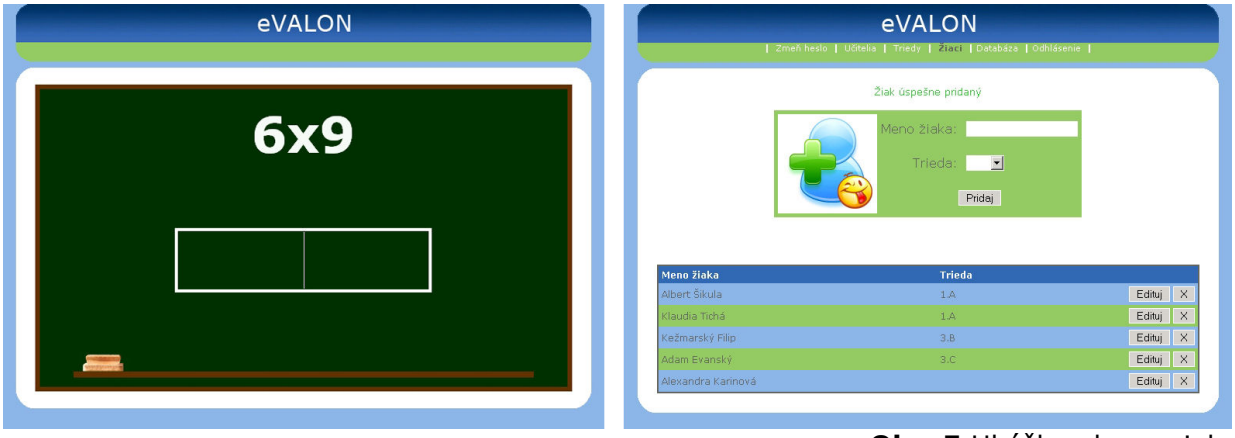


Obr. 3 Grafická reprezentácia štruktúry rozloženia

Obr. 4 Reálna grafická reprezentácia štruktúry

Pre vizuálne prvky systému boli vybrané ukľudňujúce farby, ktoré by ani pri dlhom čase nemali dráždiť zrak. Ako celok má dizajn pôsobiť jednoducho, minimalisticky aby neboli používatelia sústredený naň, ale na informácie, ktoré sú im podávané (obrázok 5 vpravo).

Prostredie pre cvičenia má pripomínať jednoduchú tabuľu bez zbytočných efektov či prílišných detailov, ktoré by mohli rozptyľovať pozornosť žiaka. Písmo zadaného príkladu ako aj vstupu je veľké a kontrastné vzhľadom na podklad, aby bola čitateľnosť čo najlepšia (obrázok 5 vľavo).



Obr. 5 Ukážka obrazoviek

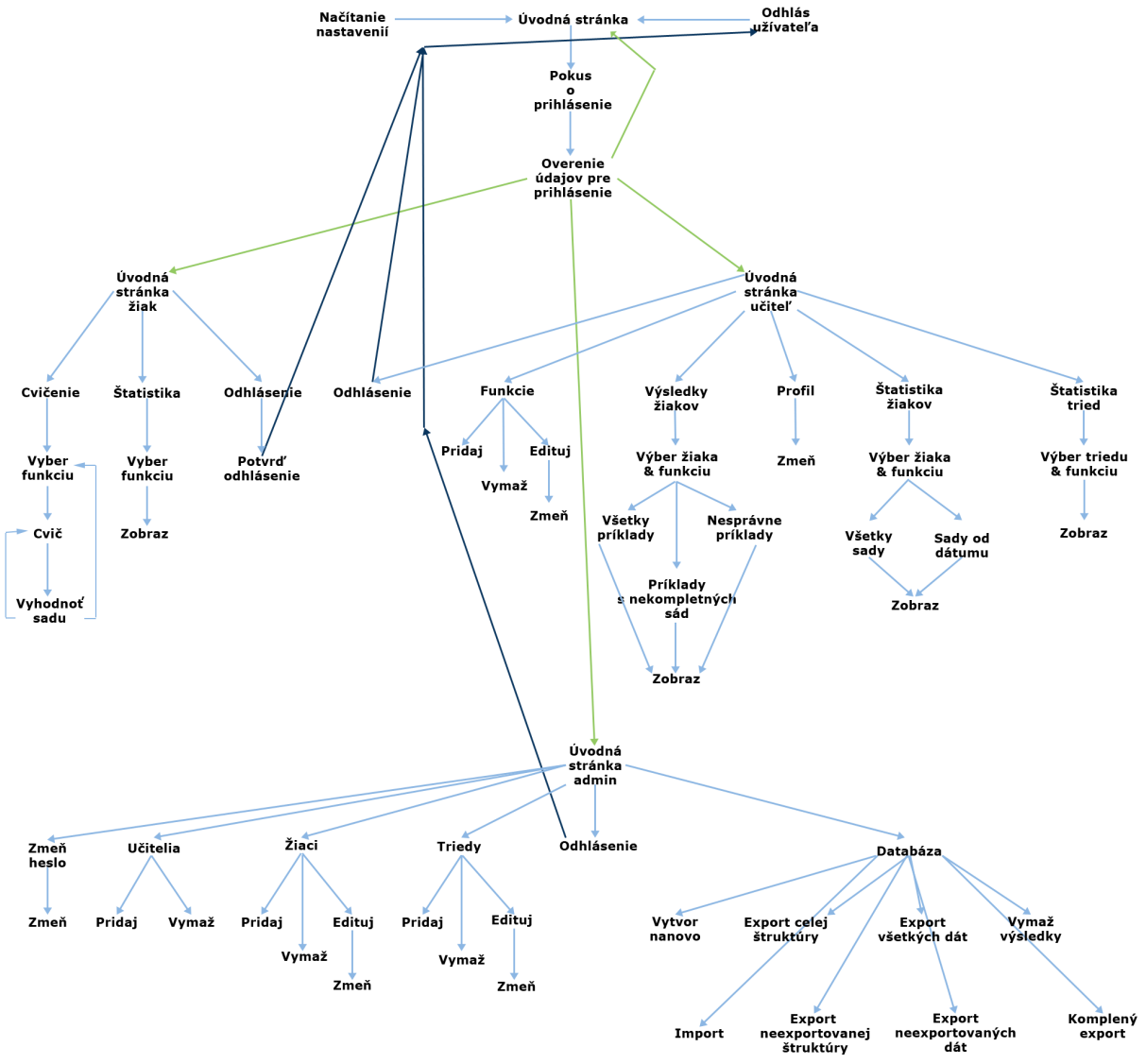
Ďalším prvkom, ktorý sme implementovali v rámci dizajnu sú veľké výrazné ikony na každej stránke s nejakou funkčnosťou ako mnemotechnická pomôcka (obrázok 6).

Obr. 6 Príklad použitých ikoniek

Funkcionálne prvky

Funkčnosť programu je rozdelená do 23 stránok a ďalších 26 scriptov. Logicky je program rozdelený do 4 celkov podľa užívateľa, ktorý ich má obsluhovať – administrátorská, učiteľská, žiacka a spoločná časť.

Obrázok 7 ukazuje zjednodušene všetky funkcie aj so vzťahmi medzi nimi.



Obr. 7 Schéma funkcii programu a ich vzájomných vzťahov

Administrátorská časť

Táto časť je určená pre osobu, ktorú budem ďalej nazývať administrátorom systému. Administrátor vykonáva úkony spojené s inštaláciou programu, administráciou užívateľskej štruktúry a databázy.

Inštalácia sa okrem skopírovania programu do koreňového adresára internetových služieb (DocumentRoot) vykonáva spustením a vyplnením setDB.php skriptu. Tento skript naplní súbor data.php informáciami o názve školy, lokácii serveru, mena oprávneného užívateľa MySQL databázy a jeho hesla. Tieto údaje sa neskôr využívajú na prístup k databáze a ich použitie nie je viditeľné zo strany klienta. Skript setDB.php okrem získanie týchto informácii vytvorí databázu, tabuľky, dve funkcie a užívateľa admin, cez ktorého pristupuje administrátor k svojím nástrojom. Tomuto kontu je ako predvolené nastavené heslo admin.

Pri administrácii užívateľskej štruktúry sa jedná o štandardné úkony ako vytvorenie, vymazanie, editovanie učiteľa, triedy a žiaka.

Medzi funkcie nad databázou, ku ktorým má administrátor prístup cez moje rozhranie, patrí vymazanie výsledkov, vytvorenie databázy nanovo, import a viacero možností pre export.

Pri exporte je využívaný atribút tabuliek export. Ten má v sebe uloženú jednu z troch hodnôt, ktoré reprezentujú tri stavy:

1. Stav s hodnotou 0 – Základný stav, ktorý naznačuje, že daný údaj ešte nebol exportovaný.
2. Stav s hodnotou 1 – Túto hodnotu majú nastavené všetky údaje, ktoré boli už aspoň raz exportované.
3. Stav s hodnotou 2 – Všetky údaje, ktoré sú vyexportované majú nastavenú túto hodnotu.

Pri importe je u všetkých údajov prestavená hodnota 2 na 1. Pri sadách a príkladoch je ešte pred touto zmenou modifikovaný identifikátor sady s_id a to tak, že sa najskôr nájde maximálne s_id už uložených sád a to sa pripočíta k práve importovaným.

Učiteľská časť

Primárne slúži učiteľom na monitorovanie výsledkov žiakov. Pre túto činnosť je pripravené širšie spektrum nástrojov, ktoré by mali sprostredkovať práve tie informácie, ktoré učiteľ vyžaduje. Učiteľ má prístup len k informáciám o triedach kde je evidovaný ako triedny učiteľ. Jedinou výnimku je štatistika pre všetky triedy, kde má možnosť porovnať svoju triedu s ostatnými. Medzi zatiaľ implementované možnosti patria:

- **Štatistika tried**
    - Pre konkrétnu triedu a funkciu – Tabuľka všetkých žiakov evidovaných v danej triede s informáciami o počte počítaných sád, chýb na sadu, úspešnosťou v percentách, priemerný čas na sadu a príklad. Pre lepšie porovnanie je na konci uvedený aj priemerný žiak triedy.
    - Pre všetky triedy a konkrétnu funkciu – Prakticky ide o prehľad priemerných žiakov vo všetkých kategóriách uvedených vyššie.
- **Štatistika žiaka**
    - Pre všetky príklady daného typu – Po zvolení tejto možnosti sa zobrazí učiteľovi tabuľka a 3 grafy (ak sú vypočítané aspoň dve sady daného typu). Tabuľka obsahuje informácie o úspešnosti, priemernom čase na sadu, úspešnosti v percentách, počte správnych a nesprávnych odpovedí. Prvý graf je koláčového typu a graficky reprezentuje úspešnosť. Druhý graf je čiarového typu a ukazuje počet správnych odpovedí pre sadu od prvej po poslednú. Tretí graf je tiež čiarového typu a reprezentuje celkový čas pre sadu od prvej po poslednú. Keďže oba posledne spomínané grafy sú čiarové s rovnakými údajmi na horizontálnej osi (očíslované sady) tak je možné na nich sledovať súvis medzi celkovým časom sady a správnymi odpoveďami v nej. Celkovo sa očakáva, že druhý graf bude mať stúpajúcu tendenciu (početnosť správnych

výsledkov vzhľadom na sadu stúpa) a tretí klesajúcu (čas na výpočet sa znižuje).
    - Pre príklady daného typu počítané od konkrétneho dátumu – táto možnosť je totožná s predchádzajúcou až na fakt, že pri sa berú do úvahy len sady počítané od zadaného dátumu.
- **Výsledky žiaka**
    - Všetky – Tabuľka všetkých počítaných príkladov obsahujúca informácie o výraze, počte správnych a nesprávnych odpovedí, ich pomer, a priemerný čas na výpočet daného príkladu.
    - Len nesprávne – Tabuľka podobná predchádzajúcej až na fakt, že sú tu vyobrazené len príklady, v ktorých žiak pochybil a pridaná je história výsledkov od prvého po posledný.
    - Z nekompletných sád – Ide o sady, ktoré neboli dopočítané z dôvodov použitia funkcie naspäť (back) čí histórie (history) webovského prehliadača. Snažili sme sa tieto funkcie nejako obmedziť, ale keďže sú implementované priamo v samotnom prehliadači a nie v kóde nie je na ne dosah. Existujú niektoré triky, ktoré vedia opticky oklamať užívateľa, ale tie nie sú použiteľné pre tento prípad, lebo by obmedzovali funkčnosť programu. Preto sme sa radšej rozhodli riešiť tento problém takým spôsobom, že žiak síce môže použiť tieto funkcie s pocitom, že nezanechal stopu (takáto sada sa nepočíta do štatistiky, ktorá sa mu ukazuje), ale zanechal stopu v podobe nulovej hodnoty celkového času sady, pretože tento atribút sa vypočíta a uloží až po dopočítaní všetkých príkladov v sade. Takto sa dajú tieto sady vyhľadať. Zobrazujeme ich do tabuľky s kompletnou históriou príkladov a výsledkov. Z týchto údajov je možné vyčítať či ide o ojedinelí prípad, spanikárenie; a na základe týchto informácii môže učiteľ zareagovať.

Ako bolo spomenuté pri štatistke o žiakovi sú použité grafy. Tieto sú vykreslené pomocou javascriptovej knižnice PlotKit. Pre toto riešenie som sme sa rozhodli, pretože je možné ju využívať pod BSD licenciou a

nevyžaduje žiadne dodatočné nastavovanie serveru ako by bolo potrebné napríklad pri použití PHP knižnice na vykresľovanie grafov JpGraph.

Toto riešenie má jednu nevýhodu a tou je slabšia kompatibilita s webovským prehliadačom Internet Explorer. Spracovanie grafov pod týmto programom je pomalšie a niekedy sú koláčové grafy nekompletné.

Ďalšia možnosť, ktorú majú učitelia k dispozícii je manažment funkcii. Ide o možnosť ich vytvárať, mazať a premenúvať. Zatiaľ program podporuje len funkcie, ktorých členy ako aj výsledok sú celé čísla. Každá z funkcii má 5 parametrov, ktoré ju definujú:

1. Meno
2. Aritmetický výraz – V svojej podstate môže učiteľ napísať ľubovoľný aritmetický výraz, kde členy sú označené ako veľké písmeno A a funkcie majú syntax, ktorá je používaná v PHP, napríklad * pre násobenie.
3. Minimálna hodnota – Pre členy A.
4. Maximálna hodnota – Pre členy A.
5. Počet príkladov v sade

Jediný parameter, ktorý je možné modifikovať po vytvorení funkcie je meno. Zmena ostatných by viedla ku skresleniam.

Žiacka časť

Žiak má na výber z menu tri možnosti – cvičenia, štatistika a odhlásenie.

Odhlásenie ako pri jedinom užívateľovi je potrebné potvrdiť. Pri oboch vyššie uvedených užívateľov sme predpokladali určitý stupeň uvedomelosti pri výbere, kde kliknú myšou, a tak sa odhlásia priamo bez medzistránky.

Štatistika pre žiaka je obmedzená verzia tej, ktorú má k dispozícii učiteľ. Dôvodom je rozdielny cieľ, na ktorý je určená. Zatiaľ čo pre učiteľa slúži na monitorovanie procesu cvičenia u žiaka je použitá na jeho motiváciu. Preto obsahuje len tabuľku a dva grafy. Tabuľka obsahuje informácie o priemernom čase na výpočet sady, úspešnosť, počet správnych a nesprávnych odpovedí. Prvý graf je koláčového typu a graficky reprezentuje úspešnosť. Druhý je čiarový a zobrazuje celkový čas na sadu od prvej po poslednú.

Najzaujímavejšou, čo sa programovania týka, bola časť na precvičovanie žiaka. Ako prvé po zvolení tejto položky sa zobrazia funkcie, z ktorých si jednu žiak vyberie.

Od chvíle výberu funkcie zastrešuje priebeh cvičenia jeden PHP script pupil_pracice.php, ktorý si ešte podľa potreby načítava pomocné PHP skripty.

Na ovládanie a celkovú interakciu s programom v procese cvičenia stačí numerická časť klávesnice. Je to z toho dôvodu, lebo používanie myši či iných ovládacích prostriedkov by mohlo spôsobiť skreslenie celkového času na výpočet príkladu. Teraz, keď je celé ovládanie na tak malej ploche a bez potreby presného zameriavania na objekt, s ktorým sa má interagovať, sú oneskorenia minimalizované. Druhým dôvodom je možnosť rozptýlenia žiaka, keby bola použitá na ovládanie myš.

Pre zaručenie tejto črty sme použili DOM funkciu focus(), ktorá je vyvolaná akciou onLoad v tagu body pri každom načítaní skriptu.

Na osvetlenie zvyšku čo sa deje na pozadí cvičenia je dole uvedený zjednodušený pseudokód tohto skriptu, za ktorým nasleduje pseudokód generovania príkladov respektíve reprezentácia jeho základnej myšlienky:

---

pupil_practice.php

```
<body OnLoad="document.form.focusElement.focus()>
//nastav pozornosť na objekt na ktorom má byť nastavená pozornosť

if (atribút začiatok počítania nastavený){
   vytvor pole pre spätnú väzbu
   //pole je vytvárané tak, že sa prejdú príklady tejto funkcie,
   //ktoré žiak v minulosti počítal a v prípade, že atribút correct
   //je rovný F0, teda ešte nebol použitý v spätnej väzbe, tak ho
   //zaradí do poľa
   premennú môžeš použiť spätnú väzbu=TRUE
   vytvor prázdne pole na zaznamenávanie príkladov, ktoré sa
     vygenerujú
   zaznač o aký druh funkcie ide aj so všetkými jej parametrami
   vytvor počítadlo príkladov a nastav ho = 0

   <table>
         Pripravený?
         <tlačidlo>Štart</tlačidlo>
         //po potvrdení sa skript znovu načíta s nastaveným
         //atribútom "Štart" a atribútom "Počítaj"
   </table>
}
if (atribút počítaj){
   if (atribút štart){
         vytvor novú sadu a ulož si jej vygenerovaný identifikátor
   }

   zavolaj funkciu na generovanie príkladu
   //táto funkcia bude opísaná nižšie
   vygenerovaný príklad vlož do poľa použitých príkladov
   zmeň príklad na vizuálnu podobu
   ulož aktuálny čas do premenej ČAS
   <table typ=tabuľa samodokončovanie_textu=vypnuté>
   //je dôležité vypnúť samodokončovanie, lebo v opačnom prípade sa
   //webovský prehliadač snaží vyhadzovať pomôcky v už napísaných
   //vstupov
         vypíš príklad
         <textarea>Miesto na písanie výsledku</textarea>
         //po potvrdení výsledku pomocou klávesy enter sa skript
         //znovu načíta a pošle žiakov výsledok ŽV, príklad P,
         //premennú ČAS a nastav atribút "Vyhodnoť"
   </table>
}
```

```
if (atribút vyhodnoť){
   čas na príklad ČP= aktuálny čas – poslaná premenná ČAS
   zvýš počítadlo príkladov o 1
   zmeň príklad P na funkčnú podobu
   správny výsledok SV = funkcia vyhodnoť(príklad P)

   if (počítadlo príkladov==počet príkladov na sadu) {
         nastav atribút "Počítaj"
   } else {
   nastav atribút "Celkové Vyhodnotenie"
   }

   if (správny výsledok SV==žiakov výsledok ŽV) {
         premenná správny S==T
         premenná text TXT=super
         premenná obrázok IMG=správne.jpg
   } else {
         premenná správny S==F0
         premenná text TXT=nabudúce to bude lepšie
         premenná obrázok IMG=nesprávne.jpg
         ulož príklad do poľa pre spätnú väzbu
   }

   ulož príklad do databázy s položkami (indentifikátor sady,
     príklad P, žiakov výsledok ŽP, správny S, čas na príklad ČP)
   <table>
         vypíš premennú TXT
         <tlačidlo>Pokračuj</tlačidlo>
         //po potvrdení sa skript znovu načíta pričom už sme vyššie
         //nastavili atribútom "Celkové hodnotenie" alebo atribút
         //"Počítaj"

   </table>
}
if (atribút celkové hodnotenie) {
   premennú môžeš použiť spätnú väzbu=TRUE
   vynuluj počítadlo príkladov
   vynuluj pole pre zaznamenávanie príkladov

   vyhodnoť príklady sady a ulož si medzivýsledky
   ulož celkový čas sady z medzivýsledkov do databázy
   podľa počtu správnych odpovedí urči výsledný obrázok {
         >10% nesprávnych: premenná IMG=guru.jpg
         <50% správnych: premenná IMG=super.jpg
         >10% správnych: premenná IMG=strelnica.jpg
   }

   <table>
         vypíš hodnotenie sady na základe uložených medzivýsledkov
         vykresli obrázok z premennej IMG
         <tlačidlo>Ďalšia sada</tlačidlo>
         //po potvrdení sa skript znovu načíta s nastaveným
         //atribútom "Štart" a atribútom "Počítaj"
         <tlačidlo>Koniec</tlačidlo>
         //po potvrdení otvor okno s výberom funkcii
   </table>
}
```

---

pupil_giveEx**.php

```
premenná príklad P=“”
minimálna hodnota MinH = minimálna hodnota MinH*2-1
maximálna hodnota MaxH = maximálna hodnota MaxH*2
//hodnoty min a max sú rozšírené aby sa zvýšila šanca vygenerovania
//nižších a stredných hodnôt, po vygenerovaní čísla sa toto vydelí
//dvomi aby sa znovu vrátilo do požadovaného intervalu

if ((môžeš použiť spätnú väzbu=TRUE)AND
    (pole pre spätnú väzbu nie je prázdne)AND
    (náhodne vygenerované číslo z možností 1 alebo 0 sa rovná 0)
   )
   // ak boli všetky podmienky splnené ide sa program pokúsiť vybrať
   // príklad z poľa spätnej väzby
{
   cyklus WHILE, ktorý testuje, či je v poli spätnej väzby nejaký
    príklad, ktorý ešte nebol použitý v aktuálnej sade:
   1. príklad sa našiel {
         v databáze nastav atribút correct pri príklade na F1, čo
           znamená, že už bol použitý v spätnej väzbe
         odstráň príklad z poľa spätnej väzby
         script sa končí – príklad, ktorý sa našiel je použitý a
           vyvezený ako príklad P
   }
   2. príklad sa nenašiel {
         môžeš použiť spätnú väzbu=FALSE
         vygeneruj náhodný príklad P
         cyklus WHILE, ktorý kontroluje či už nebol príklad použitý
           v aktuálnej sade a či je výsledkom príkladu prirodzený
           koreň:
         1. podmienky sú  splnené {
              script sa končí – príklad, ktorý sa vygeneroval je
                použitý a vyvezený ako príklad P
         }
         2. podmienky sú nesplnené{
               vygeneruj náhodný príklad P
         }
   }
} else {
   //v tejto ceste je generovaný náhodný príklad
   vygeneruj náhodný príklad P
   cyklus WHILE, ktorý kontroluje či už nebol príklad použitý
    v aktuálnej sade a či je výsledkom príkladu prirodzený koreň:
   1. podmienky sú  nesplnené {
         vygeneruj náhodný príklad P
   }
   2. podmienky sú splnené{
         príklad P= vygenerovaný príklad
   }

   if (príklad P medzi príkladmi v poli spätnej väzby) {
         v database nastav atribút correct pri príklade na F1, čo
           znamená, že už bol použitý v spätnej väzbe
         odstráň príklad z poľa spätnej väzby
   }
}
```

Zabezpečenie bezpečnosti

Systém pri svojej činnosti pracuje s množstvom údajov. Väčšina týchto údajov sa ráta medzi osobné alebo citlivé. Preto sme museli zabezpečiť ich ochranu a zariadiť aby prístup k nim mali len oprávnené osoby. Okrem dát bolo treba v systéme riadiť prístup k jeho funkcionalite.

Obe tieto úlohy sú riešené pomocou autentifikácie a jej kontroly.

Riešenie problému je implementované v troch skriptoch, patriacich do časti spoločnej pre všetkých užívateľov.

Prvý z nich zabezpečuje overovanie údajov pri prihlasovaní. V prípade správnosti presmeruje na úvodnú stránku daného typu užívateľa a nastaví premenné meno, identifikátor užívateľa, užívateľské miesto. Pre užívateľa nie je nikdy viditeľný a údaje sú skriptu posielané metódou post. Túto implementáciu sme zvolili, aby sme minimalizovali rozhranie predávania údajov, a tým ohraničil možné pokusy o nedovolený prienik do systému.

Druhým je skript na kontrolu prístupu na stránku. Ináč povedané, tento skript kontroluje či má daná požiadavka na prístup ku stránke dostatočné práva. Pri kontrole sa využívajú premenné nastavované pri prihlasovaní. Týmto sa zabraňuje pokusom o vniknutie do častí určených len prihláseným užívateľom v prípade, že k nim poznáme adresu, no nie sme prihlásení.

Posledný skript zabezpečuje bezpečné odhlásenie – vymažú sa všetky premenné aby sa nedali neskôr zneužiť.

Ďalším prvkom ochrany údajov je kontrola všetkých vstupov od užívateľa, s výnimkou administrátora, na možnú SQL insertion. Ide o techniku, pri ktorej posielané údaje obsahujú prvky SQL jazyka. Tieto by v prípade neošetrenia mohli byť programom vnímané ako chcené a boli by vyhodnotené, čím by útočník mohol docieliť viacero úkonov, napríklad prihlásenie na cudzie konto.

Pri podobných systémoch je zvykom pred uložením hesla do tabuľky užívateľa na toto heslo použiť nejakú jednosmernú hašovaciu (hash) funkciu, napríklad MD5 alebo SHA-1. Použitím hašovacej funkcie sa

dosahuje, že takto ošetrené heslo nie je priamo čitateľné. Ale je možné porovnávaním toto heslo nájsť, odhaliť v zmysle, že rovnaké heslá majú rovnakú hašovaciu hodnotu. Aby sme zabránili tejto možnosti, je pri vytvorení užívateľa náhodne vygenerovaný reťazec 10 znakov, zreťazený s heslom a až následne je takýto reťazec hašovaný. Toto riešenie má za následok zníženie šance rovnakého hašu pri rovnakých heslách a tým sa stáva autentifikácia silnejšou.

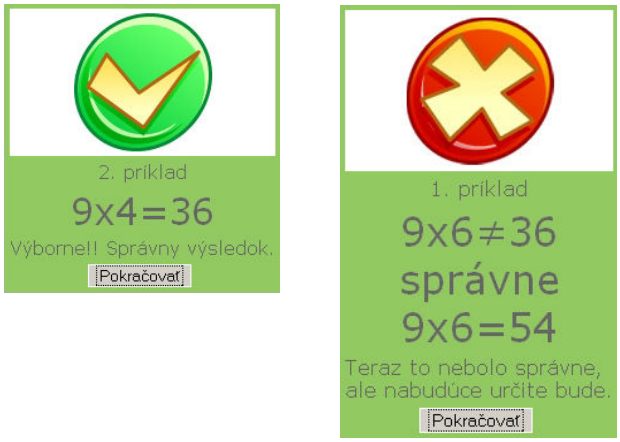
III. Výhody systému

Úspešný proces osvojovania si matematiky je založený na trvalom opakovaní, utvrdzovaní a systematizácii matematických poznatkov a ich dôkladnej náväznosti. Bez primeraného zvládnutia predchádzajúceho učiva je nemožné postupovať ďalej. Skutočnosti, ako je veľký počet žiakov v triedach a minimálny čas na individuálne pôsobenie a v nemalej miere ešte stále veľký rozsah učiva, kladú na vyučovanie matematiky vysoké nároky.[1]

Z uvedeného vyplýva hneď niekoľko výziev a problémov pri vyučovaní matematiky, s ktorými by náš systém mohol pomôcť. Medzi ne by sme rátali opakovanie a utvrdzovanie poznatkov a to aj vzhľadom na to, aké sú triedy nehomogénne prostredia. Z vlastných skúseností vieme, ako ťažké bolo pre učiteľa zosúladiť potrebu tých čo počítali rýchlo a tých čo počítali pomalšie. Myslíme si, že mnohokrát riešenie tohto problému znevýhodnilo jednu či druhú skupinu. Výhoda programu je, že žiak počíta svojím tempom toľko príkladov koľko zvláda. Silnejší nie sú brzdený pomalšími a pomalší nemusia skákať k ďalšiemu príkladu len preto, že ostatný ho už vyriešené.

Ďalší problém, ktorý by mohol program čiastočne pomôcť odbúrať je nepomer medzi počtom žiakov a potrebou individuálneho pôsobenia. Jednu časť riešenia ponúka vyhodnocovanie výsledkov priamo po ich vypočítaní. Žiak sa hneď dozvedá výsledok, za ktorý je buď hneď

pochválený alebo mu systém ukáže správne riešenie s povzbudením do ďalšieho počítania (obrázok 8).



Obr. 8 Ukážka vyhodnotenia priamo po vypočítaní príkladu

Ďalšou výhodou systému je spätná väzba. Program eviduje nesprávne výsledky konkrétneho žiaka a v nasledujúcej sade je 50% šanca, že pri generovaní príkladu bude vybraný taký, v ktorom sa niekedy pomýlil a nebol ešte preskúšaný. Tým sa zabezpečí to, že ak žiak sústavne robí chybu v nejakých príkladoch tie mu budú dávané častejšie na precvičenie, zatiaľ čo tie, ktoré ovláda menej častejšie.

Poslednou ale veľmi významnou časťou je možnosť pre učiteľa monitorovať výsledky, pokrok žiakov. Na monitorovanie je na výber viacero možností čím je učiteľovi umožnené selektívne vyberať informácie, ktoré pokladá za potrebné. Napríklad učiteľ využije voľbu pozerať si pri žiakovi iba nesprávne riešenia s históriou výsledkov čo môže veľa naznačiť o zmýšľaní žiaka, či sa mýli systémovo alebo len triafa.

Medzi faktory, ktoré ovplyvňujú každú ľudskú činnosť patrí motivácia. Výnimku nie je ani matematika. Motivovať sa dá rôznymi spôsobmi. Program využíva motiváciu súťažou, ktorá je veľmi účinná, ale môže mať jednu nevýhodu. Ak učiteľ častejšie využíva súťaživé formy práce, môže sa to negatívne prejaviť napríklad tým, že odradí slabých žiakov. Títo žiaci sa môžu záporne stavať nielen k jeho systému práce, ale aj k nemu samému, lebo nemajú takmer žiadnu možnosť uspieť. Dobrí žiaci zase vynakladajú malú námahu, ciele činnosti častokrát neprekračujú

nárokmi ich schopnosti. Nadmerne sa zaťažuje určitá skupina žiakov a v triede vzniká konfliktná a nespolupracujúca atmosféra. [1]

Tento prípad však pri správnom podaní programu učiteľom môže odpadnúť, keďže podľa našich skúseností na začiatku stačí žiaka len navnadiť na to, aby začal počítať a potom sa sám snaží prekonávať svoje posledné výsledky – súťaží sám so sebou. Pre podporu tejto vlastnosti je pre žiaka pripravená voľba odsledovať si svoje počítanie od prvej po poslednú sadu.

IV. Vízia do budúcnosti

Už počas písania programu nás napadali nové a nové možnosti ako ho rozšíriť či zlepšiť a ďalšie námety prichádzali od ľudí, s ktorými sme o tejto práci diskutoval. No nie všetky sa dostali do programu. Dôvodov bolo viac. Najdôležitejší bol ten, že sme nechceli, aby bol program prehustený, síce peknými, ale nepraktickými funkciami, čo by okrem iného spôsobilo ťažšiu ovládateľnosť a mohlo by odvádzať od samotnej podstaty programu.

Preto najlepším riešením je, aby sami učitelia povedali, bo by ešte chceli v programe doplniť, modifikovať či odstrániť. Napríklad by sme radi vedeli, ako by riešili prihlasovanie žiakov, ktoré je v súčastnej verzii programu riešené len cez meno. Implementácia prostredníctvom hesla sa nám nezdá dobrá a druhé riešenie, ktoré nás napadlo a to to, že učiteľ bude prihlasovať žiakov svojím heslo by mohlo spomaľovať prácu s programom.

V. Záver

Spracovanie bakalárskej práce výborná skúsenosť, ktorá autorovi umožnila vyskúšať si v praxi poznatky nadobudnuté počas štúdii na FMFI UK. Hlavne som mal možnosť aplikovať poznatky z oblasti webových technológii a procesu tvorby softvéru.

Podarilo sa nám úspešne splniť cieľ bakalárskej práce vytýčený v úvode, a to vytvorenie nového prostriedku, ktorý je možné začleniť do vyučovania matematiky.

Verím a dúfam, že si program si nájde cestu do praxe a pomôže žiakom ako aj učiteľom.

Vďaka tejto práci som získal množstvo skúseností, ktoré sú devízou do budúceho profesionálneho života.

VI. Prílohy

Zoznam technológii

HTML / Hypertext Markup Language

Značkový jazyk, používaný k vytváraniu formátovaných webových dokumentov. Kód sa skladá zo samotného textu a extra informácii o ňom uložených v tagoch (návestiach). Na základe týchto extra informácii prehliadač pridelí textu formát.

CSS / Cascading Style Sheets

Kaskádový štýl je jazyk používaný na opis prezentačnej časti dokumentu. Najčastejšie používaný na opis dizajnu webových stránok napísaných v HTML alebo XHTML. V poslednej dobe je presadzovaný aj ako prostriedok na štrukturalizáciu webových stránok.

PHP

Imperatívny objektovo orientovaný funkcionálny jazyk. Pôvodne navrhnutý pre potreby vývoja dynamických stránok. Má viaceré možnosti využitia, ale najčastejšie je používaný na scriptovanie na strane servera. Jazyk je vyvíjaný skupinou PHP Group a šírený pod licenciou PHP License. Free Software Foundation ho považuje za voľne šíriteľný softvér.

SQL / Structured Query Language

Jazyk vytvorený pre potreby vyhľadávania a získavania údajov z relačných databáz. Je štandardizovaný pod ISO a ANSI.

DOM / Document Object Model

Platformovo a jazykvo nezávislý štandard pre objektové modely reprezentované HTML, XML alebo podobným formátom.

Javascript

Slabo typový, procedurálny jazyk používaný na scriptovanie na strane klienta.

Zoznam použitých knižníc, skriptov a ich licencia

PlotKit

Verzia: 0.9.1

Copyright: copyright (c) 2006 Alastair Tse

Licencia: BSD License

Web: http://www.liquidx.net/plotkit/

excanvas.js

Copyright: copyright (c) 2006 Google Inc

Licencia: Apache License

Web: http://sourceforge.net/projects/excanvas/

MochiKit

Verzia: 1.3.1

Copyright: Copyright 2005 Bob Ippolito

Licencia: dual-licencia - MIT License alebo Academic Free License v2.1

Web: http://www.mochikit.com/

VII. Použitá literatúra

[1] Kubišová Eva

Ako motivujem na matematike, písomná práca k I. kvalifikačnej skúške, Čierny Balog 1996

[2] Mederly Pavol

Princípy tvorby softvéru, materiál k prednáškam

FMFI UK

[3] *Wikipedia, the free encyclopedia*

http://www.wikipedia.org