FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO V BRATISLAVE

Katedra informatiky

Systém na správu vyučovania

Diplomová práca

Bratislava 2010 Bc. Michal Hrobár

Fakulta matematiky, fyziky a informatiky Univerzity Komenského v Bratislave
Informatika

Systém na správu vyučovania

Diplomová práca

Bratislava 2010

Bc. Michal Hrobár
Vedúci diplomovej práce:
RNDr. Mária Pastorová

Týmto vyhlasujem, že som diplomovú prácu vypracoval samostatne s odbornou pomocou školiteľa, čerpajúc z uvedenej literatúry a zdrojov dostupných na internete.

Abstrakt

Jedná sa o internetovú aplikáciu (nie len) pre stredné školy, ktorá má uľahčiť vybavovanie určitej agendy a to najmä známkovanie, tvorbu a prehliadanie rozvrhov a pripravovanie suplovania. Je určená na samostatnú prevádzku na jednej škole, to znamená, že pre podporu viacerých škol treba spustiť viacero inštancií – každá škola si musí prevádzkovať svoju.

Žiacka knižka implementuje role administrátora, učiteľov, žiakov, rodičov a ostatných. Žiaci si vedia nastaviť, aké informácie môžu vidieť konkrétni používatelia.

Na automatické generovanie rozvrhu je použitý rozvrhoý software FET.

Projekt bol inšpirovaný vznikom komplexného software-u pre školy aSc Agenda a jeho komerčným nasadením.

Obsah

Úvod

Ciele

Výsledná aplikácia by mala spĺňať nasledovné vytýčené body:

- Slúžiť ako internetová žiacka knižka bežiaca samostatne na školskom serveri
- Poskytovať možnosť manuálne upravovať rozvrh
- Na automatické generovanie rozvrhov používať program FET
- Za aktívnej účasti učiteľov pomáhať organizovať suplovanie
- Šírená pod licenciou GPL
- Zdrojový kód vrátane komentárov v angličtine
- Naprogramovaná v PHP 5, bez akýchkoľvek varovných výpisov pri nastavení oznamovacej úrovne „E_ALL“
- Rešpektuje normu XHTML 1.0 Strict
- Rešpektuje normu CSS 2
- Okrem používateľskej role administrátora plne funkčná aj bez povolenia vykonávania JavaScriptu na klientskej strane
- Použitá SQL databáza: MySQL
- Použité kódovanie UTF-8

Krátky úvod do internetových aplikácií

Internetová aplikácia je program, ktorý ako svoje používateľské rozhranie používa webstránku. Používateľ sa pripojí pomocou bežného internetového prehliadača a server mu poskytne dynamicky generovanú webovú stránku, čiže stránku, ktorá dokáže meniť svoj obsah podľa okolností. Program je ovládaný nasledovaním liniek a vypĺňaním rôznych formulárov na tejto stránke.

Výhodou internetových aplikácií je ich nepretržitý beh a možnosť práce viacerých používateľov naraz bez potreby inštalovať nejaký špeciálny software na používateľský počítač. Dokonca sa používateľ nemusí ani fyzicky nachádzať pri počítači, na ktorom program beží. Stačí mať iba prehliadač a možnosť pripojiť sa k serveru.

Medzi nevýhody možeme zaradiť napríklad nemožnosť práce offline. Pri pomalom spojení môže mať používateľ tiež dojem, že program nereaguje, alebo nefunguje správne. Rovnako sa nemôžeme spoliehať, že požiadavky prichádzajúce od klienta sú "správne". Používateľ môže používať napríklad tlačítka back, forward alebo refresh, alebo napísať akúkoľvek URL sám, takže systém sa nemôže spoliehať, že používateľ (alebo útočník) vždy nasleduje iba nejakú z ponúknutých možností.

Stránku robia dynamickou najmä dva prvky: vykonávanie PHP skriptu na strane servera a vykonávanie javaskriptu na strane klienta. V dnešnej dobe síce skoro každý webový prehliadač javaskript podporuje, používateľ ho môže mať však vypnutý, alebo môže vypnúť prehliadač bez toho, aby zmeny uložil. Všetky dôležité úkony by sa teda mali vykonávať PHP skriptom. Javaskript je dobrý najmä na zvýšenie komfortu pri používaní a sprehľadnení stránky.

Súčasný stav podobných programov

Existuje veľa rôznych e-learningových programov, ako napríklad známy Moodle. Tieto sú však špecializované na výučbu, distribúciu materiálov, online skúšania a odovzdávanie úloh. Sú určené skôr pre vysoké školy. Neriešia tvorenie rozvrhu či suplovanie. Známky, ktoré študenti získavajú, sú skôr informácie pre nich ako pre ich rodičov.

Rovnako existuje viac programov na tvorbu rozvrhov. Najprepracovanejší (čo do obmedzení pri tvorbe rozvrhu) je asi FET, ktorý je šírený pod licenciou GNU/GPL. Ďalší, komrečne rozšírený, je aj aSc Timetables. Množstvo slovenských škôl tento software používa. Ďalší na Slovensku rozšírený program, je o niečo starší a má názov Doklady. Primárne je určený na spravovanie údajov o žiakoch a ich známkach a tlač vysvedčení.

Tieto programy však pracujú na jednom počítači a s učiteľmi veľmi neinteragujú. Nie je možnosť prihlásiť sa z domu v dobe, kedy by sa určite nedalo do školy dovolať a nahlásiť neprítomnosť, alebo pozrieť si rozvrh čerstvo po zmene. Programy síce umožňujú HTML export, ale to si vyžaduje prídavný čas zamestnanca, ktorý každý deň export urobí a výstup sprístupní na webe.

V poslednej dobe veľa škôl tiež prešlo na internetovú žiacku knižku, ktorá však beží na externom serveri, kam teba známky vypĺňať, čo niektorí považujú za bezpečnostné riziko.

Krátky úvod do bezpečnosti

Client-server architektúra

Client-server architektúra znamená, že databáza či aplikácia beží centralizovane na nejakom serveri a užívatelia používajú klienty, aby ju na diaľku ovládali. HTTP protokol na prezeranie webstránok funguje request/response, čiže klient zadá požiadavku napríklad vo forme adresy URL a server mu ako odpoveď zostaví a pošle webstránku.

PHP aplikácie sú teda skripty, ktoré dostanú HTTP request ako vstup a vygenerujú HTML stránku. Nebežia nepretržite, ale server ich vykoná, až po prijatí requestu. Týmto sa vynárajú základné témy súvisiace s HTTP cache, šifrovaním a označovaním si rôznych užívateľov pomocou cookies.

HTTP cache

Najjednoduchšie využitie HTTP je prenášanie stránok so statickými informáciami. Tým myslím, že k jednej URL existuje jedna stránka, ktorú má klient zobraziť a neočakávame, že by sa táto mala často meniť. Aby sa ušetrilo na sieťovej komunikácií, existujú v internete HTTP cache, ktoré si stránky ukladajú a v prípade opätovného prezerania tej istej ju poskytnú bez kontaktovania cieľového servera.

Aby sa stránky však vôbec mohli meniť, HTTP cache po nejakej dobe záznamy vymazáva. V prípade internetovej aplikácie je cacheovanie samotného obsahu stránky (čiže nie napríklad statických obrázkov) nežiadaný efekt. Práve naopak, ako používateľ nasleduje linky, alebo používa rôzne iné ovládacie prvky, chceme, aby vždy videl stránku zodpovedajúcu jeho aktuálnej činnosti.

Jedným z riešení je napríklad generovanie vždy novej adresy URL, čo by však malo za následok napĺňanie cache pamätí zbytočnými údajmi. HTTP implementuje prostriedky na informovanie cache pamätí o charaktere stránky. HTTP správa obsahuje rôzne hlavičky s informáciami o časovej platnosti stránky. Upravíme ich teda, aby cache pamäte cestou stránku neukladali. Je zvykom vyplniť nie len hlavičku zakazujúcu cacheovanie ale aj dobu expirácie na minulosť kôli starším HTTP cache.

Šifrovanie

Niekedy je potrebné preniesť údaje dôverne. Na to slúži šifrovanie, ktoré nie je predmetom tejto práce. Na šifrovanie HTTP sa bežne používa SSL/TLS, ktoré je súčasťou každného moderného web servera. Pri SSL sa šifrovaný kanál odlišuje číslom portu, TLS implementuje šeciálne správy pre začiatok a koniec šifrovaného spojenia. Princíp zabezpečenia spojenia je ten istý.

Ten spočíva v používaní symetrického šifrovacieho kľúča, ktorý sa dohodol po asymetricky šifrovanom kanále. Asymetrický kľuč servera je obsiahnutý v certifikáte, ktorý treba nainštalovať do prehliadača, alebo dať podpísať nejakej certifikačnej autorite, ktorú bude prehliadač považovať za dôveryhodnú.

V školských podmienkach nie je problém (je bežné), aby správca vytvoril vlastnú certifikačnú autoritu a jej certifikát nainštaloval na používateľské počítače.

Cookies a Sessions

Nakoľko jednotlivé zobraznia stránok aplikácie bežia samostatne, oddelene a pre viacero používateľov súčasne, treba si jednotlivých aktuálnych používateľov nejakým spôsobom rozlíšiť. Rovnako, jeden používateľ môže mať otvorených aj niekoľko okien prehliadača a pracovať v rôznych častiach systému. Jedno takéto používanie budeme volať session. Nie je vhodné na rozlišovanie session použiť IP adresu, nakoľko sa táto v internete môže aj prekladať a bolo by to aj porušenie OSI modelu. Pri začiatku session teda vygenerujeme unikátne session ID (SID) a na základe tohto budeme medzi užívateľmi rozlišovať.

Na distribúciu SID sú zaužívané (aj s podporou v PHP) dva hlavné spôsoby: pridávanie SID do URL všetkých liniek a uchovávanie pomocou cookies. Prvý spôsob má niekoľko vážnych nevýhod, preto sa viac používajú cookies. Aj v tejto práci používam cookies. Ako nevýhody by sme si mohli spomenúť napríklad nutnosť vkladať SID ako `<input type="hidden">` tag do všetkých formulárov, odosielaných metódou get, keďže všetky polia formuláru sa majú dostať do URL (a nahradiť prípadné hodnoty, ktoré sa tam nachádzali z atribútu `action`). Rovnako používateľ môže ľahko nechtiac skopírovať URL, z ktorej sa dá SID ľahko prečítať a až do odhlásenia zneužiť.

Druhý spôsob spočíva v odoslaní špeciálnej hlavičky, ktorú nám až do zrušenia/uplynutia platnosti bude prehliadač posielať naspať ako svoju identifikáciu. Tejto hlavičke sa hovorí cookie. Okrem SID si do cookie nepotrebujeme ukladať nič viac, lebo všetky ostatné údaje ostávajú uložené na serveri v session súbore, ktorého názov je sess_SID. Do session sa ukladajú napríklad údaje o prihlásení, aby stačilo meno a heslo napísať do opätovného odhlásenia len raz.

Útoku, pri ktorom sa útočník snaží zmocniť cudzieho SID, sa nazýva session stealing. Najzákladnejšia obrana je šifrovanie celého spojenia vrátane odoslania hlavičiek. Prehliadač by rovnako nemal poskytnúť cookie stránke, ktorá ho neukladala a to ani ak sa naň pýtame skriptom. Tým však celá obrana nekončí. Treba tak isto dbať, aby útočník nemohol nikam v rámci systému umiestniť skript (XSS), ktorý by potom používateľove SID odhalil. Možností, ako odhaliť SID je veľké množstvo, no väčšina pochádza najmä z nesprávneho použitia a nedostatočného zaškolenia používateľov.

Obľúbená metóda útoku, keď sa nemôžeme dostať k používateľovmu SID je napríklad session riding, kedy sa nemusíme SID dozvedieť, stačí, ak ho dokážeme použiť. Typickým príkladom môže byť napríklad kliknutie na linku z e-mailu počas práce v systéme. Ak to bola linka, po navštívení ktorej sa udejú v systéme nejaké zmeny, udejú sa s právami obete. Prísna obrana by bola napríklad generovanie jednorazových URL pre každú akciu. Zamedzila by však pohodlnému viacoknovému používaniu a používaniu tlačítok ako back, refresh, alebo bookmarkov. Osvedčuje sa aj dobrý návyk všetky zmeny v systéme vykonávať metódou post. Metóda get je vhodná na ukladanie nastavenia rôznych zobrazení a podobne. Bookmarkovanie stránky aj číslom práve vymazávaného užívateľa bude skor nežiadaný efekt.

Napokon ešte treba spomenúť, že ochranou SID chránenie session nekončí. Rovnako treba chrániť samotné údaje session na serveri. Web server si súbory jednotlivých session ukladá ako dočasné súbory. Tieto rozhodne nesmú byť čitateľné ostatnými používateľmi servera. Obvykle sa web server nastaví tak, aby jednotlivé stránky bežali s právomocami používateľa, ktorý je vlastníkom daného skriptu (suPHP). Potom treba vytvoriť pre internetovú aplikáciu samostatné používateľské konto, ktoré bude vyhradené iba pre tento účel. Session súbory potom budú čitateľné/zpisovateľné iba týmto kontom.

Pri ukončovaní session, napríklad pri odhlasovaní sa z aplikácie treba všetky príslušné údaje vymazať. Samotný súbor s údajmi treba vymazať a používateľovi poslať cookie so SID s nastaveným časom expirácie do minulosti.

SQL injection

SQL injection je metóda útoku, keď sa útočník dovtípi, čo s daným vstupom urobíme a snží sa zvoliť ho tak, aby sa systém zasprával nejako, ako nebolo pôvodne zamýšľané. Predstavme si napríklad, že autentifikáciu užívateľa robíme príkazom:

```
mysql_query("SELECT * FROM uzivatelia WHERE login='" .
      $_GET['login'] . "' AND pwd='" . $_GET['pwd'] . "'");
```

A následne akurát skontrolujeme, či sa našiel aspoň jeden taký riadok. V takom prípade stačí, aby útočník napísal ako login „' OR 1=1 OR '' = '“ a ako heslo napríklad prázdny string, a po dosadení bude teda príkaz vyzerať:

```
mysql_query("SELECT * FROM uzivatelia WHERE login='' OR 1=1
      OR '' = '' AND pwd=''");
```

a teda vďaka podmienke 1=1 vráti nejaký riadok, ak v tabuľke vôbec nejaký je. To, že sa nám dokáže niekto prihlásiť do systému bez zadania hesla ešte stále nie je to najhoršie. S takouto chybou v programe útočník nemá problém napísať do dotazu napríklad bodkočiarku a začať písať príkazy podľa ľubovôle. V lepšom prípade napríklad „delete“, na čo správca rýchlo príde a obnoví databázu zo zálohy, v horšom napríklad „update“ a nastaví si napríklad rolu administrátora. Funkcia mysql_query() síce neumožňuje napísať do príkazu bodkočiarku, ale mohli by sme takýto útok realizovať napríklad pomocou vnoreného príkazu.

Obrana proti takémuto útoku je zrejmá: treba formátovať textové reťazce tak, aby boli databázovým parserom vždy rozpoznané ako text a nie ako príkazy. Na to slúži PHP funkcia mysql_real_escape_string(). Pri programovaní treba však držať na pamäti aj bezpečnostné kroky nesúvisiace priamo SQL injectionom. Ako dobrá praktika sa ukazuje aj používanie čo najmenišieho množstva podmienok používajúcich stringy. Je vhodnejšie, aby riadky mali celočíselný primárny kľúč a na joinovanie a identifikáciu riadkov sa používal práve on. O vstupe sa oveľa jednoduchšie zistí, či je numerický, ako či je to správne formátovaný string. Rovnako pomocou čísla nie je možné urobiť injection a dá sa menej pokaziť. Pri písaní každého zásahu do databázy treba dbať na overenie údajov pred zápisom a nespoliehať sa na žiadny formát alebo vlastnosť vstupu. Jednotlivé zásahy majú byť atomické a nezávislé, čo sa s číslami implementuje omnoho bezpečnejšie z hľadiska kontroly programátorových chýb.

Bezpečnosť hesiel

Existuje mnoho spôsobov, ako autnentifikovať používateľa. Najstaršia a najtradičnejšia je asi autentifikácia heslom. Používateľ obdrží, alebo si vymyslí nejaký reťazec, za ktorý sa zaväzuje, že ho pozná iba on sám. Úskalia tejto metódy sú napríklad pri zabudnutí hesla, alebo jeho zapísaní. Používateľ si často heslo volí ľahko uhádnuteľné, alebo veľmi jednoduché – krátke alebo nad malou abecedou.

Zabudnuté heslo sa často rieši napríklad pomocou bezpečnostnej otázky, na ktorú ak používateľ správne odpovie, bude mu poskytnuté heslo. Týmto býva bezpečnosť napriek silnému heslu často znížená, lebo sila bezpečnostnej otázky je obyčajne veľmi malá. Napríklad: „meno domáceho zvieraťa“, „meno manželky za slobodna“, „názov základnej školy“... Týmto sa zároveň prezrádza, aj čo je odpoveďou a teda akú informáciu treba získať. Takéto informácie obyčajne o používateľovi vedia všetci, čo ho aspoň trochu poznajú a zároveň im otázka nechtiac prezrádza, akú odpoveď treba zadať. Keby tam otázka napísaná nebola, jednalo by sa vlastne iba o akési alternatívne heslo, ktoré sa dá rovnako dobre zabudnúť, ako to predchádzajúce.

Ďalšia možnosť získania zabudnutého hesla často býva jeho preposlanie na používateľov e-mail. Takáto možnosť si však vyžaduje dobré zabezpečenie aj mailovej schránky. Nikdy by sa nemalo užívateľovi zaslať pôvodné heslo, lebo ho môže používať na prístup do viacerých aplikácií a prípadný útočník by získal prístup aj k nim. Namiesto toho systém posiela novovygenerované aj s linkou, po kliknutí na krorú si môže používateľ rovno nastaviť nové heslo. Vygenerované veľmi náhodné heslá nad veľkou abecedou sú síce veľmi silné, ale užívatelia ich príliš neobľubujú, lebo sú im pridlhé alebo jednoducho ťažko zapamätateľné. Preto je dobré dať možnosť hneď si takéto heslo bez námahy zmeniť. Ak by bol postup komplikovaný, hrozí, že používateľ si mail ponechá a pri každom prihlásení otvorí, aby zistil heslo. To obyčajne tiež neodpisuje, ale na jeho prenesenie do prihlasovacieho formulára použije schránku a heslo po takomto prenesení ostane v nej. Môže sa stať, že sa z nej (možno nedopatrením) heslo dostane. Zároveň každý, kto získa prístup k používateľovej e-mailovej schránke, získa rovnako aj prístup ku všetkým v nej uloženým heslám.

Ukladanie hesiel a saltovanie

Na autentifikáciu heslom treba dve veci: poslať heslo na server a porovnať ho s uloženým. Netreba ho pritom posielať alebo mať uložené vždy v podobe čitateľného textu. Dá sa napríklad ukladať iba nejký jednosmerný hash. Overuje sa potom porovnaním hashov. Zabezpečenie sa potom opiera o fakt, že je ťažké nájsť kolidujúce heslá.

Treba si uvedomiť, že hashovať má zmysel iba jednu z informácií, buď uložené heslo, alebo to, ktoré sa posiela po sieti. V prípade zahashovania obidvoch by vlastne pôvodné heslo stratilo opodstatnenie, a jednalo by sa o porovnanie čistého textu.

V prípade autentifikácie do webových aplikácií sa často používa metóda posielania čistého textu a ukladania hashu, nakoľko sa na posielanie hesla najprv nadviaže šifrované SSL spojenie. Rovnako je to aj v prípade tohoto projektu. Ukladanie hashovaných hesiel do databázy pomáha chrániť heslá pri úniku databázy, prípadne chráni administrátorov proti obvineniu zneužitia ich právomoci.

Ďalšie zlepšenie tohoto spôsobu prináša saltovanie, kedy je do tabuľky loginov a hashov hesiel pridaný ďalší stĺpec salt. Salt je nejaký náhodný reťazec, ktorý sa pripojí ku heslu pred tým, než sa zahashuje. Salt sa vygeneruje pri vytovrení/zmene hesla. Použitie tejto techniky chráni databázu proti vyhľadávaniu používateľov s rovnakým/kolidujúcim heslom. Salt môže rovnako obsahovať binárne dáta, ktoré sa bežne v heslách nevyskytujú, čím výrazne rozšíri abecedu a chráni databázu pred vyhľadávaním kolízií na základe predpočítaných hashov.

Ako vygenerovať bezpečné heslo?

Silné heslo je teda nie len také, ktoré je ťažké na uhádnutie, alebo dlho trvá jeho prelomenie systematickým skúšaním všetkých, ale najmä také, ktoré spĺňa účel, čiže autentifikuje – je ľahké na napísanie pre správnu osobu a takmer neuhádnuteľné inou osobou.

Najprv si zvolíme nejký základ hesla a potom ho dodatočne ozdobíme nejakou prídavnou informáciou, aby jeho uhádnutie inou osobou bolo ešte ťažšie. Ako základ hesla je dobré si zobrať niečo, na čo si bez problémov spomenieme, keď sa budeme chcieť autentifikovať k danej službe. Tento základ by bolo dobré zvoliť tak, aby len málokomu napadol taký istý. Napríklad ako základ k heslu k bankovému účtu nie je vhodné použiť slovo „banka“ alebo „účet“. Oveľa efektívnejšie je napríklad spomenúť si na niečo, čo sa nám osobne asociuje so slovným spojením „bankový účet“, napríklad sme si ako deti prvýkrát otvorili účet s vkladom 100 korún a v tom čase sme mali prezývku Janko Hraško.

V takom prípade môžeme dostať napríklad heslo „J1krn00H“, ktoré sa nám pamätá extrémne ľahko, lebo prvé a posledné písmeno sú iniciálky našej prezývky a do čísla 100 sme akurát umiestnili slovíčko „korún“ bez samohlások, no pre cudziu osobu sa zdá úplne chaotické. Ďalšou inšpiráciou môže byť napríklad naša obľúbená fráza napísaná na inej klávesnici (napríklad v azbuke) foneticky prepísaná a skrátená naspäť našou, alebo fonetický prepis obľúbenej pesničky v cudzom jazyku odzadu. Fantázii sa medze nekladú.

Generátor hesiel

Pre tých, ktorým sa heslo nechce ťažko vymýšľať, ponúka systém na stránke jednoduchý nástroj, do ktorého používateľ napíše slovo či frázu, ktorá sa mu ľahko pamätá a systém mu ponúkne niekoľko náhodne odvodených variant ako kandidátov na silné heslo. Ako parametre sa dajú nastaviť:

- dĺžka hesla – koľko slabík pôvodnej frázy sa použije, koľko znakov ešte nástroj pridá a nahradí...
- veľkosť abecedy
    - na akej klávesnici chce užívateľ zadávať heslo
    - chce používať kláves SHIFT?
    - aké netradičné znaky je ochotný písať (čísla, interpunkčné znamienka...)
    - obľúbenosť čísel - niektorí ľudia majú dobrú pamäť na čísla a radi si ich dávajú do hesla. Tento parameter nahradí niekoľko alfabetických znakov numerickými. Výsledné heslá však budú dlhšie.
- anatomické rozpoloženie – pre heslo je rovnako dôležité, aby sa aj ľahko písalo
- vysloviteľné – pre niektorých používateľov je dôležité, aby sa heslo dalo vysloviť

Pri anatomickom rozpoložení nástroj zohľadňuje aj pozície znakov na kávesnici. Ak bude heslo pohodlné na napísanie, bude sa dať napísať rýchlejšie a bude ťažie na odsledovanie prípadným prizerajúcim sa. Rovnako človek už nebude uvažovať nad jednotlivými znakmi, ale bude ich zadávať ako bloky, ku ktorým si vyvinie akýsi vzťah, podobne ako klavíristi nehrajú jednotlivé tóny ale akési harmonické celky alebo akordy.

Ďalšou vlastnosťou nástroja je farebné rozlíšenie slabík zadanej frázy. Výsledné heslá majú rovnaké zvýraznenie písmen, čo tiež pomáha ako mnemotechnická pomôcka. Pri tvorbe hesla sa uprednostňuje použitie celej slabiky, najme jej začiatku. Ak používateľ napríklad zadal ako vzorku slovo "Nástenka", oveľa ľahšie sa mu zapamätá napríklad podpostupnosť znakov "nás" alebo "ka" ako napríklad "ást" alebo "nk".

Všetky vygenerované heslá slúžia iba ako inšpirácia, takže v prípade, že používateľa žiadne heslo nezaujme, môže si vymyslieť vlastné, alebo jednoducho zmeniť parametre a vygenerovať si ďalšiu sériu. Vygenerované heslá sú však veľmi pestré a náhodné obsahujúce nie len časti frázy a preto sú ťažké na uhádnutie skúšaním aj keď poznáme zadanú frázu.

Implementácia generátora hesiel

Silu hesla alebo algoritmu, ktorý ho vygeneroval, posudzujeme podľa veľkosti množiny hesiel, ktoré daný algoritmus generuje.

Existuje niekoľko rôznych druhov generátorov hesiel. Medzi najbežnejšie patrí vygenerovanie nejakej náhodnej postupnosti znakov konkrétnej dĺžky. Pre danú dĺžku je to zároveň najlepšia možnosť z hľadiska sily hesla. Pre ľudí je však príjemné zapamätať si radšej aj niekoľko znakov naviac, no mať ho po určitej stránke jednoduchšie. Po akej stránke má byť heslo jednoduché, aby sa ľahko pamätalo je celkom ťažká otázka. Niektoré generátory napríklad majú oddelené množiny samohlások a spoluhlások a heslá vyrábajú ich náhodným striedaním, prípadne pridaním nejakých predpôn a prípon, čím vytvárajú vysloviteľné heslá. Takto vygenerované heslá však musia mať aj dvojnásobnú dĺžku, aby boli rovnako silné ako náhodné heslá.

Generátor hesiel implementovaný v systéme vyžaduje nejaký vstup od používateľa – počiatočnú frázu, na ktorú si ľahko spomenie a ktorá mu následne bude heslo pripomínať.

Uľahčí to aj pravidelné menenie hesla. Pri použití rovnakej počiatočnej frázy sa budú heslá podobať. Či je to vítaný efekt, sa už rozhodne používateľ.

Poradie krokov

Pri generovaní hesla postupujeme nasledovne:

1. Počiatočná fráza sa rozdelí na slová a jednotlivé slová na slabiky. Následne sa vynechajú jednopísmenné slabiky, ktoré môžu platne vzniknúť napríklad pri slove „a-hoj“. Ich prítomnosť by nám technicky nevadila, ale krátke slabiky znižujú počet možných nagenerovaných hesiel a rovnako nie sú veľmi dobrou mnemotechnickou pomôckou.
2. Niekoľko slabík vyberieme a zreťazíme. Môžu sa aj opakovať. Nové reťazce budeme vkladať len medzi slabiky.
3. Z výsledného reťazca vyberieme niekoľko znakov na rôznych pozíciách a zameníme ich za vygenerované náhodné vzorky podľa nastavnení.
4. Vložíme dlhší reťazec čísel. Jeho predvolená dĺžka je 0. Ak má používateľ pamäť na čísla, môže nastaviť dlhšiu.
5. Vložíme ďalšie náhodné vzorky.
6. Vložíme prípadné samohlásky, aby bolo heslo vysloviteľné
7. Na niektoré znaky uplatníme stlačenie klávesu shift.

Rozdelenie na slabiky

V prvom kroku algoritmus rozdelí slovo na slabiky. Nie vždy sa mu to podarí presne podľa slovenského pravopisu, ale to nie je veľmi dôležité. Použije sa nasledovný algoritmus:

1. vyhľadajú sa samohlásky (všetky znaky, čo nie sú vymenované ako spoluhlásky považujeme za samohlásky – zadaná fráza nemusí byť skutočné slovo, môže to byť hocijaký reťazec, ktorý sa používateľovi ľahko pamätá)
2. ak sa nejaké R alebo L nenachádza vedľa samohlásky, tiež sa považuje za samohlásku
3. samohlasky sa očísľujú z ľava do prava – za sebou idúce samohlásky dostanú to isté číslo (z hľadiska vytvárania hesla je pre nás užitočné nedeliť ich každú zvlášť)
4. spoluhlásky na začiatku a na konci slova sa očíslujú rovnako ako prvá a posledná samohláska
5. neočíslované spoluhlásky, ktoré majú napravo samohlásku sa k tejto priradia
6. neočíslované spoluhlásky, ktoré majú naľavo samohlásku sa k tejto priradia (je dôležité zachovať toto poradie)
7. všetky ostatné spoluhlásky sa priradia k samohláske pravo

Čísla priradené k písmenám určujú, do koľkátej slabiky patria. Počas celého algoritmu treba ku „ch“ a „dz“ pristupovať ako k jednému písmenu. O každom znaku výsledného hesla si počas generovania pamätáme, z koľkátej slabiky pochádza (číslované od 0). Prídavné vygenerované znaky budú patriť slabike -1.

Tento jednoduchý algoritmus zafunguje na veľa slovenských slov, ako napríklad „že-le-zo“ a pri väčšom množtsve spoluhlások prvú priradí predchádzajúcej slabike „že-lez-ný“. Rovnako prihliada na dvojhlásky a skupiny spoluhlások: „sta-ni-čiar“, „nad-štrb-ska“. „L“ a „R“ vedľa samohlásky nie sú slabikotvorné: „maš-kr-ta“, „stĺ-po-vý“, „tla-čo-vé“. Dokonca si poradí aj so slovom „Ka-rl-ton“. Slabikotvorným je v tomto prípade „L“.

Niektoré (zvášť cudzie) slová však nepodlí správne. Náročná implementácia by však v tomto prípade bola zbytočná, nakoľko písaním medzier sa dá slovo podeliť aj ručne. Príklady nesprávneho algoritmického delenia: „Chor-vát-sko“, „a-rach-no-fó-bia“.

Generovanie výplne

Na výber sú 2 možnosti generovania jedného znaku výplne:

- náhodne zvolíme nejaký znak z klávesnice
- zvolíme dva znaky, ktoré sa na klávesnici nachádzajú blízko pri sebe (anatomické generovanie)

Základná abeceda výplne sa skladá z písmen A-Z, rozšíriť sa dá napríklad o čísla, interpunkčné znamienka na klávesnici vpravo a znaky vo vrchnom riadku klávesnice. Je teda dôležité poznať aj rozloženie kláves na klávesnici, ktorú chce používateľ použiť.

Pri anatomickom generovaní znakov zase obetujeme dĺžku hesla na úkor vzdialenosti znakov na klávesnici. Vyberieme miesto, kam vložíme výplň, a jeden zo susedných znakov pomenujeme kontext. Namiesto jedného náhodného znaku teraz vložíme najviac dva, ktoré sa na klávesnici nachádzajú blízko kontextu.

Čo to znamená blízko? Najjednoduchšie riešenie je zobrať štvorec susedov 3x3 okolo kontextového znaku. Týmto získame 9 znakov, z ktorých jeden alebo dva vyberieme. Množina sa dá ešte zväčšiť, ak pri výbere prvého znaku priberieme do množiny na vyberanie aj vštkých jeho susedov. Blízko by v takomto prípade znamenalo, že spomedzi kontextu a dvoch nových znakov sa dá vybrať jeden, ktorý na klávesnici susedí s ostatnými.

Problém však nastáva, keď sa kontext nachádza na kraji klávesnice. V takom prípade by množina susedov na výber bola výrazne menšia a klesal by aj počet možných vygenerovaných hesiel. Ďalším problémom môže byť napríklad nerovný okraj klávesnice (klávesy nie sú zarovnané do obdĺžnika) alebo dokonca diery v nej (niektoré znaky nie sú vhodné do hesla, napríklad samotný znak mäkčeň). Za takýchto podmienok môže byť umiestnenie štvorca 3x3 tak, aby obsahoval kontext, problematické.

Preto použijeme inú metódu: klávesnica je reprezentovaná ako mriežka a na vybratie „susedov“ použijeme prehľadávanie do šírky. Rovnako ako pri štvorci, vyberieme niekoľko kandidátov a ešte susedov prvého vybraného znaku – tentokrát nám už nevadí, keby neexistovali. Po výbere dvoch znakov sa ešte rozhodneme, v akom poradí ich napíšeme.

Ako prehľadávanie naozaj funguje je opísané v časti Detaily (strana 17).

Príklad

Uveďme si príklad opäť s použítím počiatočnej frázy „Nástenka“ – všetky rozhodnutia sú len príklady a robíme ich náhodne:

1. nájdeme samohlásky (á-e-a) a rozdelíme slovo na slabiky – Nás-ten-ka
2. vyberieme dve z nich a vložíme medzi ne výplň – ka?ten
3. bola vybraná možnosť anatomickej voľby výplne, rozhodli sme sa vložiť na miesto otáznika dva znaky. Vybrali sme „s“ a „x“, lebo sú blízko posledného znaku minulej slabiky – „a“.
4. výsledkom je heslo „kasxten“, ktoré sa naozaj ľahko píše.

`	1	2	3	4	5
	Q	W	E	R	T
	A	**S**	D	F	G
	Z	**X**	C	V	B

Tabuľka 1: Časť klávesnice

Modifikácia pôvodných znakov

Aj samotné slabiky by bolo vhodné pred použitím ešte mierne zmodifikovať, aby výsledné heslá nemali zbytočne dlhé spoločné podreťazce a tým nemrhali dĺžkou. Jednoduchšie ako pred výberom slabík je to však až s výsledným reťazcom.

Takže náhodne vyberieme niekoľko znakov z doteraz vyrobeného hesla, ktoré však pochádzajú z pôvodných slabík a nahradíme ich znakmi zvolenými rovnakým postupom ako pri tvorbe výplne. Ako kontext zvolíme nahrádzané písmeno, alebo

Vloženie čísel

Do výsledného reťazca môžeme vložiť aj dlhší číselný reťazec.

Vysloviteľnosť

V prípade, že treba, aby bolo heslo nazáver aj vysloviteľné, stačí aby v ňom nenachádzala skupina viac ako troch neslabikotvorných spoluhlások za sebou. Prítomnoť iných ako alfabetických znakov si volí užívateľ, takže v prípade ich nevysloviteľnosti ich odstráni. Stačí teda použiť rovnaký postup na vyhľadanie samohlások ako v prvom kroku a do veľkých skupín spoluhlások navkladať samohlásky, aby neboli viac než 3 po sebe.

Použitie SHIFTu

Nakoniec sa už len uplatní, či chce používateľ stláčať aj shift. Ak chce, podľa voľby sa vyberie maximálne niekoľko znakov a zmení sa im case. Je dôležité uvedomiť si, že shift môže byť potrebné použiť napríklad, keď ho používateľova klávesnica vyžaduje na písanie čísel, alebo keď dané znaky obsahovala rovno počiatočná fráza.

Pri tomto kroku treba zároveň dať pozor, aby sme číselnému reťazcu case nemenili.

Detaily

Algoritmus celý čas pracuje so stringami kódovanými v UTF-32. Výhoda tohoto kódovania je, že sa v ňom dájú reprezentovať akékoľvek znaky, ktoré používateľ zadá a tiež má fixnú dĺžku jednotlivých znakov, čo uľahčuje delenie na písmená.

Pri vyhľadávaní samohlások používame tri množiny: slabikotvorné spoluhlásky, spoluhlásky a ostatné znaky. Znak, ktorý nie je spoluhláska považujeme za samohlásku, aby aj špeciálne znaky ako zátvorky a podobne vytvárali samostatné slabiky a nereťazili sa do zbytočne dlhých slabík.

Podpora iných jazykov a klávesníc sa do programu dodáva iba v podobe ďalšej funkcie na deľbu slabík a mapy klávesnice.

Pri anatomickom generovaní výplne sa špeciálne berie ohľad na kláves shift pre kontextový znak. Je pohodlnejšie nemeniť stav stlačenia shiftu pri písaní. Napríklad na anglickej klávesnici sa číslice píšu bez shiftu, na slovenskej so shiftom. Ak teda bude treba

vygenerovať výplň v kontexte „6“, na slovenskej klávesnici bude jedna z možností napríklad „TG“. Pri anglickej klávesnici by to bolo malé „tg“.

Pri výbere druhého znaku výplne zaradíme do množiny na výber aj prázdny reťazec. Tým prirodzene náhodne vyberieme počet znakov a žiadna z možností nebude nastávať pravdepodobnejšie.

Na prehľadávanie klávesnice použijeme prehľadávanie do šírky. Nezohľadňujeme teda skutočnú vzdialenosť klávesov, ale iba počet krokov, na ktorý sa z kontextového znaku vieme k damému znaku dostať. Mohli by sme implementovať napríklad best-first-search, ktorý by uvažoval skutočnú vzdialenosť klávesov, ale to pri takých malých počtoch nie je dôležité. Posledné zlepšenie vnesie úvaha, že klávesy na klávesnici nie sú rozostavené do mriežky, ale skôr ako šesťuholníky. Pri vymenovávaní susedov zaradíme teda iba 6 okolostojacich klávesov. Náš algoritmus prehľadáva najprv vo vodorovnom potom vo „zvislých smeroch“. To sa hodí pri kontexte na kraji klávesnice, kedy pri písaní používame krajné prsty a je pohodlné ich pri písaní striedať. Do fronty sú klávesy okolo „0“ zaradené v poradí, ktoré znázorňuje Tabuľka 2.

		3	5	
	1	0	2	
		6	4	

Tabuľka 2: Poradie zaraďovania do fronty

4	5	6	7	
	R	**T**	**Y**	U
D	**F**	**G**	**H**	
	C	**V**	**B**	N

Tabuľka 3: 7 susedov klávesu „G“

Tabuľka 3 ukazuje klávesy vybrané ako blízke klávesu „G“. Algoritmus vždy zaradí všetky susedné klávesy a zastaví sa, až keď dosiahne (niekedy teda aj prevýši) stanovený počet – v našom prípade 7.

Porovnanie sily generovaných hesiel

Pre používateľa je tiež dôležité vedieť správne odhadnúť silu svojeho hesla. Mnohé stránky ukazujú silu napísaného hesla počas toho, ako ho používateľ zadáva. Berú pri tom do úvahy napríklad jeho dĺžku, či obsahuje čísla alebo nejaké ďalšie znaky, čím ho vlastne nútia zvoliť si heslo nad dosť veľkou abecedou, či sa mu to páči, alebo nie.

Poďme si porovnať zopár mohutností množín hesiel s nejakými spoločnými vlastnosťami. Začnime uistením si pár faktov: znakov A-Z je 26, cifier 0-9 je 10 a znakov hore a na boku klávesnice je (pre jednoduchosť počítania) 20. Vyjadrime si, koľko je 6 znakových hesiel nad alfanumerickými znakmi: $36^6 = 2\,176\,782\,336$. Niektoré stránky sú menej náročné na používateľov. Vyžadujú síce 6 miestne heslo, no stačí, ak v ňom majú jedno číslo a jeden špeciálny znak. Vygenerujeme teda 4 znakové heslo a vložíme doň patričné znaky: $26^4.(5.10).(6.20) = 2\,741\,856\,000$. Čiže približne rovnako veľa možností. 6 miestnych hesiel zložených z písmen čísel aj šeciálnych znakov vpravo je $46^6 = 9\,474\,296\,896$.

Vzorec pre náš algoritmus má vo všeobecnosti dolný odhad na počet hesiel:

$$N \geq \binom{s+v}{v}.n^v.\binom{s.2}{m}.(n-1)^m = \frac{\prod_{s<i\leq s+v} i}{v!}.n^v.\binom{s.2}{m}.(n-1)^m$$

kde n = veľkosť abecedy, s = počet slabík, v = počet vkladaných znakov, m = počet modifikovaných znakov. Je teda zrejmé, že počas generovania, sa vždy musia najprv vybrať pozície, kam sa umiestnia/modifikujú znaky a až potom sa nahradia, aby nedochádzalo k častejšiemu generovaniu niektorých hesiel (keď si náhodou vyberieme na modifikáciu tú istú polohu). Dvojka vo vzorci vystihuje skutočnosť, že slabika musí mať aspoň dva znaky. To je aj požiadavka generátora – zadaná fráza musí mať aspoň dva znaky.

Dolný odhad je to pre to, lebo sme doň nerátali skutočnú dĺžku slabík, výber a usporiadanie slabík (predpokladali sme len jednu dvojznakovú), obohatenie o samohlásky, aby bolo heslo vysloviteľné a rozširovače abecedy ako kláves shift či anatomické generovanie hesla.

Anatomické generovanie hesla rozšíri abecedu z 26+20 symbolov (26 alfabetických, 20 kláves v hornom rade a na boku klávesnice, kam patria aj čisla) na aspoň 56 symbolov (7*7 dvojznakových + 7 jednoznakových). Je dobré všimnúť si, že výberom prvého znaku sa môžu možnosti výberu druhého znaku rozšíriť, viď Tabuľka 1 – vďaka pridaniu „s“ sme mohli pridať aj „c“. Táto úvaha navyše platí iba v prípade, že by sme používali ten istý znak ako kontext. V bežnom prípade budú však znaky okolo vkladanej pozície rôzne (a vyberať medzi nimi budeme náhodne), čo tiež môže abecedu iba rozšíriť.

Pridanie možnosti používať kláves shift zvýši počet možných hesiel ešte 2^d krát, kde d = dĺžka výsedného hesla, čiže podľa doterajšieho označenia minimálne $2^{s.2+v}$ krát.

Pridanie čísel tiež akurát prospeje, rovnako pridávanie samohlások, aby heslo bolo vysloviteľné.

Pre konkrétnu predstavu si môžeme vypočítať, koľko rôznych hesiel nám môže algoritmus nagenerovať pre jednoslabičnú počiatočnú frázu, napríklad „ba“, s parametrami: slabiky = 3 („bababa“), vložiť = 3, modifikovať = 2, abeceda alfanumerická (n=36) bez obľuby čísel, anatomického generovania, použitia shift alebo vkladania ďalších samohlások:

$$N = \binom{3+3}{3}.36^3.\binom{3.2}{2}.(36-1)^2 = 17\,146\,080\,000$$

čiže približne rovnako veľa (2x viac) ako 6 miestnych hesiel nad 46 znakovou abecedou. Program pre zadané parametre vypočíta aj dolný odhad počtu rôznych hesiel a zobrazí logaritmus tohoto čísla s rôznymi základmi pre porovnanie dĺžky rovnako silných hesiel nad rôznymi abecedami.

Používateľské role

Každý používateľ má v systéme svoje prihlasovacie meno (login), heslo a reálne meno. Žiaci majú navyše triedu, do ktorej patria a učitelia triedy, ktoré vyučujú.

Administrátor

Administátor je používateľ, ktorý môže so systémom robiť všetko, čo ponúka. Zaujímavá otázka pri tejto roli je politika pri vymazávaní administrátorov. V postate sú na výber dve možnosti: Administrátor môže vymazať iba svoj administrátorský účet, alebo naopak – iba účty ostatných administrátorov. Na čo je ktorá politika dobrá?

Prvá je bezpečnejšia z hľadiska ukradnutia prihlasovacích údajov. Získaním administrátorského účtu sa útočník nemôže stať jediným správcom systému, avšak vymazanie takéhoto konta sa dá spraviť iba manuálnym zásahom do databázy. Hrozí tiež, že administrátor si vymaže konto nedopatrením, a aj keby mal systém kontrolu, či existuje aspoň jeden administátor, ešte vždy si operátor nemusí pamätať heslo.

Druhá alternatíva je použitá aj v projekte. Nepredpokladá sa, že by administrátori úmyselne konali vo vzájomný neprospech, a nemožnosť vymazať si vlastné konto zaručuje, že vždy existuje nejaké administrátorské konto, na ktoré sa dá prihlásiť. Keď sa však útočníkovi podarí získať administrátorské práva, môže sa stať neobmedzeným vládcom a systém môže zachrániť opäť iba manuálny zásah do databázy.

Administrátor je jediná rola, ktorá dokáže pridávať a rušiť kontá učiteľom a žiakom, zaraďovať ich do tried, pripravovať rozvrh a podobne.

Učiteľ

Učiteľ v systéme môže vypĺňať známky žiakom, ktorých učí, z predmetu, ktorý ich učí.

Špeciálne triedny učiteľ generuje žiackym kontám rodičovské heslá, ktoré sú potrebné pri podpisovaní známok. Tieto potom nejakým spôsobom doručí rodičom. Napríklad na rodičovskom združení. Rovnako vidí všetky známky. Nemá zmysel, aby nevidel, nakoľko on je ten, kto rodičovské heslá generuje.

Žiak

Je v záujme systému ochraňovať osobné údaje, preto si každý žiak môže vybrať, či sa má pri výpise žiakov triedy zobrazovať ako číslo alebo ako meno. Rovnako si môže vybrať, či bude jeho známky vidieť každý, iba prihlásení, alebo iba vymenovaní používatelia systému.

Iný

Každý používateľ má uložené meno a helso v tabuľke používateľov. Rola je v nej uložená iba pre jednoduchosť. Dala by sa zistiť napríklad joinom s tabuľkou triednej príslušnosti žiaka či učiteľa. Táto rola je však jediná, ktorá si dokáže sama vytvoriť či zrušiť konto. Ostatné role majú v systéme ďalšie zviazané údaje a vytára ich administrátor.

Mať konto je potrebné, aby žiaci mohli nastavovať jednotlivým používateľom prístup k ich známkam, alebo zobrazeniu mena na stránke. Používateľ v roli „iný“ môže zadať rodičovské heslo od žiackeho konta a tým sa stať jeho rodičom. Ako rodič bude vždy vidieť

všetky informácie o žiakovi a bude môcť podpisovať jeho známky. Systém nemá špeciálny typ konta pre rodiča, aby nebránil učiteľovi mať zároveň v škole aj svoje dieťa.

Principiálne umožňuje byť rodičmi aj žiakom navzájom, čo však nijako nevadí. Na získanie rodičovského stavu je potrebné zadať rodičovské heslo, ktoré by pri prezradení nebolo problematické zneužiť s novovytvoreným kontom.

Učitelia zároveň budú vidieť, ktoré známky boli kým a kedy podpísané. Podpísať znamená iba potvrdiť prehliadnutie. Niečo ako označenie mailu ako prečítaného.

Žiacka knižka

V štandardnom zobrazení bez prihlásenia je vidieť organizovaný zoznam tried, ako ho navrhol administrátor. Zobrazené sú v tabuľke – pôvodne zamýšľanej na ročníky krát štúdijné zamerania / školy, ale použiteľné na ľubovolné grafické podelenie. Do jednej bunky sa dá umiestniť aj viac tried. To je vhodné pre triedy ako „3.A“, „3.B“, „3.C“. Zvlášť, keď číslo v označení nevypovedá jednoznačne o ročníku. Napríklad by sa teda mohli ocitnúť v jednej bunke triedy: „1.Ag“, „9.B“, „Kvinta C“, „V.Ao“ a pod.

Po vybratí triedy sa zobrazí zoznam žiakov a ich známok podľa ich nastavení súkromia.

Žiacke rozhranie

V žiackej knižke si každý žiak môže pozerať svoje známky. Rovnako nastaviť, ktorí používatelia budú môcť vidieť jeho známky a meno.

Zobrazené sú všetky známky naraz, v tabuľke podľa predmetov a období (mesiace, „polročné vysvedčenie“, ...).

Učiteľské rozhranie

Rodičovské heslá

Špeciálne triedny učitelia (trieda môže mať ľubovoľné množstvo triednych učiteľov) dokážu žiakom svojej triedy vygenerovať rodičovské heslá. Toto je jediný údaj, ktorý potrebujeme na asociovanie sa ako rodič s nejakým žiakom, preto sa nezadáva ručne, ale je náhodne generovaný automaticky a pre každého žiaka unikátny.

Pri asociácií netreba zadávať ani meno žiaka, preto by sa zdalo, že útok skúšaním všetkých hesiel je ľahko realizovateľný. Nakoľko sú zoznamy žiakov aj tak verejne dostupné (minimálne ich čísla ID), stačilo by systematicky skúšať heslá všetkým. Počet pokusov by teda bol v najhoršom prípade rovný súčinu počtu hesiel a žiakov. Ak teda systém vyhľadáva vo všetkých žiakoch naraz, stačí znásobiť počet možných hesiel počtom žiakov, aby bolo útok rovnako zložité spraviť, akoby sme každé heslo skúšali na všetkých žiakoch. Aj tak obyčajne chceme zistiť heslo konkrétnemu žiakovi.

Povedzme si, aké ťažké heslo chceme od jedného žiaka a potom ho už len patrične predĺžme. S obľubou sa na takéto účely používa napríklad 32 bitové číslo zapísané v 16kovej sústave, čiže 8 znakov 0 až 9 a A až F. Použime teda číslo 2^{32} = 4 294 967 296 ako referenčný počet hesiel na žiaka. Náhodné heslá zostavíme z veľkej abecedy a čísel. Aby bolo overovanie pohodlnejšie vynecháme z týchto symbolov ešte 14 takých, ktoré by sa mohli popliesť pri odpisovaní: {D, O, Q, 0}, {U, V}, {B, 8}, {I, 1}, {Z, 2} a {S, 5}. Dostaneme 22 znakovú abecedu. Pri heslách tvaru ACE-FGH-JKL dostávame teda 22^9 možností – pri našom počte na žiaka, by ich teda škola mohla mať 6183, čo bohato stačí.

Známkovanie

Učitelia v pohľade na žiaka vedia prejsť do editačného módu, kde si vyberú obdobie a môžu meniť známky z predmetov, ktoré ich vyučujú. Vidia pri tom aj verziu, ktorú rodič naposledy podpísal. Pri triednom zobrazení si vyberajú predmet a obdobie. Obdobia boli do systému pridané iba na podporu akýchsi termínov, rozlíšenie hlavných druhov hodnotenia a zamedzeniu dodatočného menenia známok. Inak je najjednoduchšie vytvoriť iba jedno obdobie.

Pri vypĺňaní známok sa používa tradičné jednoriadkové textové pole alebo ponuka s možnosťami – to záleží od charakteru obdobia. Obdobie „koncoročné vysvedčenie“ spravidla obsahuje iba jednu známku. Napríklad obdobie „domáce úlohy“ môže obsahovať viac známok. Jednotlivé známky sa oddeľujú medzerami.

Na rozlíšenie druhov známok je vhodný prvok štýlovania známok. Štýl známok je preklad z ich textovej reprezentácie, ako ich vypĺňa učiteľ, na grafickú, ako sa budú zobrazovať ostatným v systéme. Štýly nastavuje administrátor, takže konkrétne nároky bude môcť na požiadanie splniť. Popri vypĺňaní známok JavaScript hneď zobrazuje aj výsledok. Toto vnáša do systému vlastné druhy známok, ako napríklad za domáce úlohy, rôzne malé či veľké písomky, plusky atď. Je však už na vnútornej politike školy, či všetci učitelia budú používať štýly uniformne, alebo každý podľa svojeho uváženia. Jedna naštýlovaná známka môže mať napríklad menovku „mc1“ a zobrazovať sa ako malá čierna jednotka. U jedného pedagóga môže však znamenať známku z päťminútovky u iného za dobrovoľnú prácu.

Preklad sa realizuje pomocou statického nahrádzania označení známok výsledným HTML textom s možnosťou využitia (a prepínania) vlastného externého CSS štýlu. Systém obsahuje aj základné preddefinované stýly. Nakoľko sa jedná o veľmi zriedkavú zmenu ale častú potrebu čítania, nie sú štýly uložené v databáze ale v konfiguračnom PHP a CSS súbore. PHP súbor má syntax definície poľa. Podľa tohto súboru sa vygeneruje aj client-side JavaScript. Štýlovanie je vhodné použiť napríklad, keď majú triedy rôzny známkovací systém, napríklad na Slovensku rozšírený „1, 2, 3, 4, 5“ a „A, B, C, D, E, F“

Učitelia môžu v prípade núdze pripísať medzi známky aj textovú poznámku, no známkové tabuľky to výrazne zneprehľadňuje. Odporúčam na každú príležitosť vytvoriť radšej vlastný štýl alebo obdobie.

Rodičovské rozhranie

Po kliknutí na „registrovať sa ako rodič“ budeme vyzvaní na zadanie rodičovského hesla. Ak zadáme nejaké existujúce, overí sa ešte jeho časová platnosť. Ak je v poriadku, žiak sa nám zaradí do zoznamu detí a tento sa po našom prihlásení zobrazuje nad zoznamom tried. Pri prezeraní známok svojeho dieťaťa je k dispozícii potvrdzujúce tlačítko na podpísanie známok. Známky sa podpisujú iba všetky naraz, neočakáva sa, že by ich bolo tak veľa, že by ich rodič na jeden krát neprehliadol všetky. Podpisovanie slúži len na informovanie učiteľov.

Ak heslo už nie je platné, vypíšeme patričnú hlášku. To nepredstavuje žiadne bezpečnostné riziko, nakoľko k uhádnutiu iných hesiel to nijako nepomôže, na asociáciu so žiakom už neposlúži a na obnovenie platnosti sa musí vygenerovať nové.

Po uplnynutí platnosti sa heslo už nedá použiť na vytvorenie rodičovskej relácie. Pri bežnom použití sa rodič asociuje s dieťaťom iba raz – po prvom rodičovskom združení. Ostatné pokusy sú skôr snahou žiakov podpísať si nepríjemnú známku. Počas platnosti hesla sa však môže asociovať aj viac účtov. Záleží už od konkrétneho usporiadania v rodine.

Ostatní rodinní príslušníci sa môžu pridávať a prezerať aj sami – bez rodičovského hesla. Samozrejme so žiakovým zvolením.

Administrátorské rozhranie

Kôli jednoduchosti (aj bezpečnosti) jednotlivé skripty overujú právomoci tesne pred zobrazením nejakého komponentu. Ak je prihlásený administrátor, jednoducho je vždy oprávnený daný komponent vidieť, prípadne sa nahradí takým, že v ňom dokáže vykonávať aj zmeny. Navyše sa mu zobrazujú niektoré komponenty a menu položky navyše.

Napríklad v zobrazení žiackej knižky má prostriedky na konfiguráciu jednotlivých známkovacích období. Pri zobrazení zoznamu tried nimi dokáže hýbať v rámci ročníkov a študijných zameraní. Atď.

Databázové pozadie

Aké údaje si pamätáme pre jednotlivé entity vystupujúce v časti žiacka knižka? O všetkých používateľoch si pamätáme plné meno a e-mail. Nebudem ich teda extra menovať.

Triedy

Pre triedy je v systéme iba jedna tabuľka. Stačí si pamätať názov, skratku a polohu triedy v zozname (riadok, stĺpec). Názov sa vyskytuje napríklad ako nadpis nad známkami alebo rozvrhom, skratka vystupuje v tabuľkách.

Poloha je určená jednoducho číslami, aby sme ako číslo riadku mohli použiť napríklad očakávaný rok maturity a tým mať všetých rovesníkov v jednom riadku. Rovnako posunutie všetkých tied o riadok nižšie nebude treba vôbec uvažovať. Jednoducho pridaním nového ročníka sa nové miesto vytvorí. Rovnaký princíp aplikujeme aj na stĺpce reprezentujúce napríklad zameranie štúdia, aby sa akákoľvek trieda dala hocikedy bez námahy vybrať spomedzi ostatných, bez vytvárania dočasného stĺpca.

Nasledovný príkaz vypisuje triedy sa vynecháva prázdne riadky a stĺpce:

```
SELECT t1.x, t2.y, classes.name
FROM
(SELECT DISTINCT x FROM classes) AS t1
INNER JOIN
(SELECT DISTINCT y FROM classes) AS t2
LEFT JOIN classes ON classes.x=t1.x AND classes.y=t2.y
ORDER BY t2.y, t1.x, classes.t;
```

Príkazy ako presúvanie celých stĺpcov sú naďalej pohodlné vďaka hromadnému označovaniu a manipulácií s triedami.

Žiaci

O každom žiakovi si pamätáme:

- triednu príslušnosť
- rodičovské heslo
- dátum a rodiča posledného podpísania známok
- viditeľnosť mena a známok pre jednotlivé role (žiakov, učiteľov a neprihlásených)
- extra tabuľku pre viditeľnosť pre konkrétnych používateľov

Učitelia

O učiteľovi si musíme pamätať zoznam tried v ktorých učí a v ktorých je triednym. Počet nie je obmedzený.

Známkovacie obdobia

Nakoľko rôzne študijné smery môžu mať rôzne termíny a názvy pre spoločné skúšky, treba obdobia viazať s každou triedou zvlášť. Vrámci triedy sa však už nešpecializujú na

jednotlivé predmety a nemôžu si ich vytvárať sami učitelia, lebo pohľad na známky z rôznych predmetov (prípadne celej treidy) by bol značne neprehľadný.

Do databázy teda uložíme:

- odkedy dokedy sa v období môžu meniť známky
- pri známkovaní ktorých tried sa má zobrazovať
- počet známok: 1 (aj možnosti) alebo viac – vhodné pre známky na štatistické účely ako štvrťrok, vysvedčenie...

Známky

Všetky známky pre jedno obdobie a predmet si pamätáme ako string. Zobrazujeme ho podľa štýlov rozoberaných v časti Známkovanie (str. 22). Táto podoba je lepšia ako pamätať si každú známku osobitne kôli jednoduchosti. Učiteľ môže známky meniť, alebo dokonca vymazávať a stále vidí verziu, ktorú naposledy videl rodič. Rovnako používateľské rozhranie je omnoho jednoduchšie, ako mať rôzne prvky pre upravovanie vkladanie a vymazávanie známok, lebo pri prevedení na text a späť nie je jednoznačné, ktoré známky sú vlastne nové a ktoré snáď iba zmenili polohu (a majú ostať podpísané).

Takto jednoducho pri podpísaní skopírujeme aktuálne známky do naposledy videných. Zdalo by sa, že ukladať dátumy poslednej zmeny a podpísania je redundantné, avšak v praktickom živote zaujímavé. Posledné podpísanie je uložené v kontexte celého žiaka – stačí ukladať poslednú zmenu.

Tvorba rozvrhu

Nároky na rovrh

Rozvrh je CSP (constrain satisfaction problem). Hlavné obmedzenia pritom sú, aby jeden učiteľ neučil viac hodín v tom istom čase, aby jedna trieda nemala v tom istom čase viac hodín a aby neprebiehalo viac hodín naraz v tej istej učebni. Na problém sa môžeme pozerať ako na akési farbenie grafu, kde vrcholy sú jednotlivé hodiny, ktoré treba do rozvrhu umiestniť. Každý vrchol má práve dve farby: čas a miestnosť. Hrana vedie medzi hodinami, ktoré zdieľajú učiteľa alebo triedu. Vrcholy spojené hranou nesmú mať rovnaký čas. To by znamenalo, že trieda alebo učiteľ by mali byť v tom istom čase na rôznych miestach. Žiadne dva vrcholy nesmú byť ofarbené rovnakou kombináciou farieb. (Nesmú byť v tej istej miestnosti naraz dve vyučovacie hodiny.)

Celá tvorba rozvrhu však nie je len taká jednoduchá. Niektoré hodiny môžu zaberať aj viac miest v rozvrhu, musia sa konať v špeciálnej miestosti alebo nemusia prebiehať každý týždeň.

Iný ľahšie predstaviteľný pohľad na problém môže byť ukladanie tehličiek do viacrozmerného celočíselného priestoru. Rozmermi v tomto prípade sú čas (deň krát hodina), miestnosť, vyučujúci a trieda. Obvyklé rozvrhy napríklad pre konkrétnu triedu získame projekciou hotového umiestnenia všetkých tehličiek z podpriestoru danej triedy do časovej osi. V rozvrhu sa potom nemôže nachádzať viac tehličiek pre žiadny podpriestor určený časom a miestnoťou (resp. časom a triedou alebo časom a učiteľom).

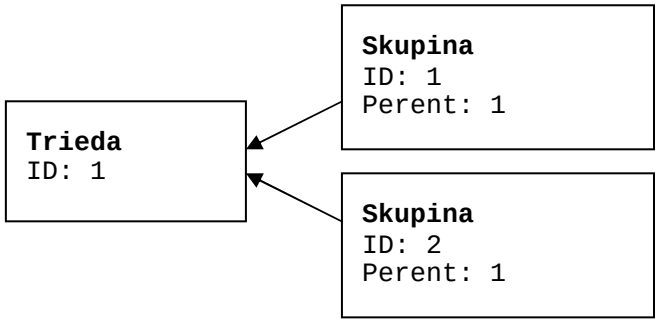
To sú len úplne najzákladnejšie požiadavky na tvorbu rozvrhu. Takisto si možno ľahšie predstaviť rôzne zložitejšie požiadavky ako: „Trieda musí začínať deň konkrétnym predmetom“, „Trieda musí/nesmie mať voľnú hodinu“, „Učiteľ nemôže učiť v istý čas“ a pod...

Delenie na skupiny

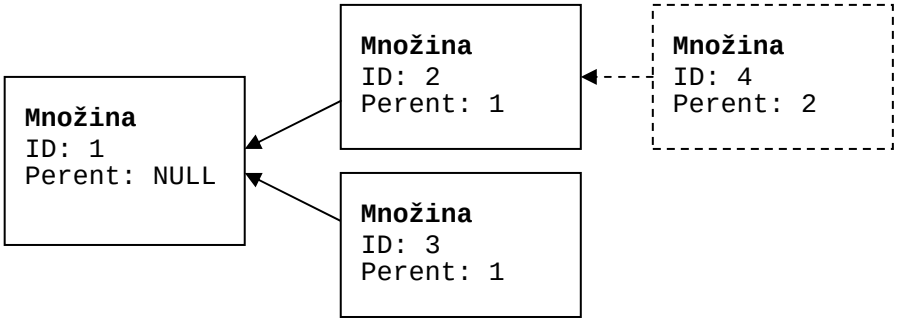
Problematika

Dôležitá súčasť rozvrhu je delenie tried na skupiny. Problematické pri tom môže byť tieto skupiny interpretovať, a tiež overovať podmienky, či dané dve hodiny nekolidujú.

Prirodzený spôsob, ktorý by napadol ako prvý, je udržovať triedy a skupiny v dvoch rôznych tabuľkách. Jedna skupina by pritom patrila vždy práve jednej triede. Takto by sme mohli rozdeliť triedu na ľubovoľné nezávislé skupiny a umiestňovať ich do rozvrhu. Toto riešenie je nepraktické iba v tom, že pri umiestňovaní hodiny do rozvrhu by sme mali na výber z dvoch tabuliek a bolo by potrebné si pamätať, do ktorej sa daná hodina odkazuje.



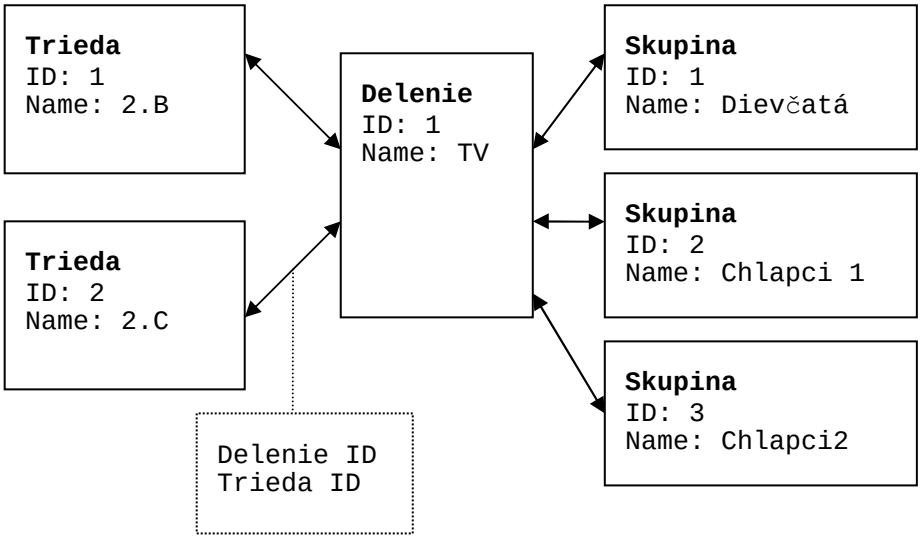
Mierne zlepšenie by mohol priniesť spôsob, kedy si budeme skupiny aj triedy ukladať do tej istej tabuľky, ktorej pridáme akurát o stĺpec naviac – príslušnosť k nejakej triede. Ak sa jedná o skupinu, bude tu mať vyplnené, do akej triedy patrí a ak sa jedná o triedu, bude tu mať napríklad NULL. Tento spôsob by nám umožnil aj elegantné delenie na podkupiny. Zisťovanie kolízií pri tom by sa výpočtovo skomplikovalo, nakoľko by každá skupina mala zoznam nadskupín (nie konštantný počet), s ktorými by sa nesmela v rozvrhu vyskytovať v ten istý čas. V praxi sa tiež nevyskytujú veľmi hlboké delenia, takže nebude problém ich reprezentovať ako jednoúrovňové/máloúrovňové.



Takto by sa dala trieda rozdeliť, ale stále existujú situácie, kedy sa napríklad majú dve triedy spojiť, alebo dokonca dve triedy rozdeliť na viac skupín, pričom žiaci v nich budú premiešaní. Cieľom je, aby delenie tried na skupiny bolo komfortné. To znamená, že neprichádza do úvahy, aby operátor zrušil dve triedy, vytvoril jednu reprezentujúcu obidve a zlial dohromady skupiny, na ktoré sa delili, aby mohol pridať ďalšie skupiny. Takéto pridanie delenia na *n* skupín by v najhoršom prípade z*n*-násobilo počet pôvodných skupín, nakoľko nemôžeme predpokladať, že pôvodné skupiny sú s novými disjunktné. Rovnako by sa znepríjemnilo aj vymenovanie skupín, ktoré majú mať určitú hodinu.

Systém teda potrebuje spravovať aj spájanie tried rovnako ako ich rozdeľovanie. Napríklad, ak sa trieda delí na angličtinu začiatočníkov a pokročilých a tiež na biológov a dejepisárov, systém by nemal povoliť ľubovoľnú angličtinu zároveň s biológiou, no dejepis a biológiu môže. Musí mať teda prehľad o tom, ktoré skupiny sú disjunktné.

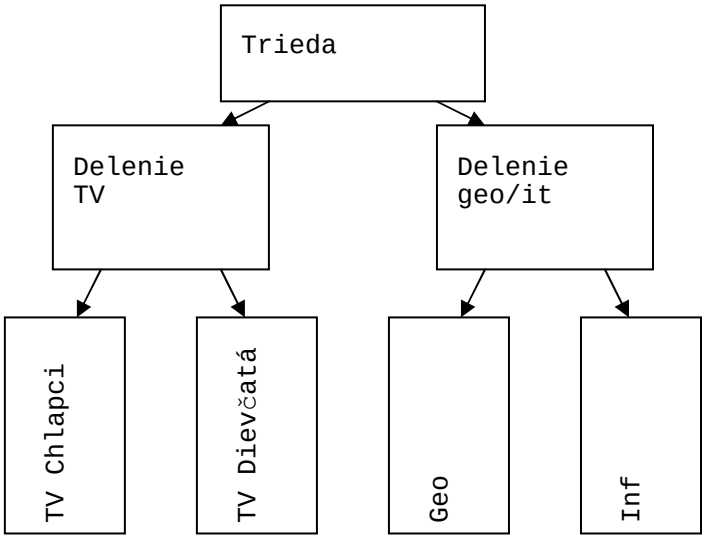
Takéto situácie sa riešia tromi tabuľkami na údaje a dvomi na relácie medzi skupinami. Jedno delenie vždy pozostáva z množiny spájaných tried a množiny disjunktných skupín. Skupiny sú to, čo sa bude zaraďovať do rozvrhu a triedy sú to, na základe čoho sa budú detekovať kolízie.



Ak v tom istom čase prebiehajú hodiny rôznych skupín, ktoré patria do rôznych delení a tieto majú spoločnú triedu, považujeme ich za kolidujúce, lebo sa nemôžme spoľahnúť, že sú disjunktné.

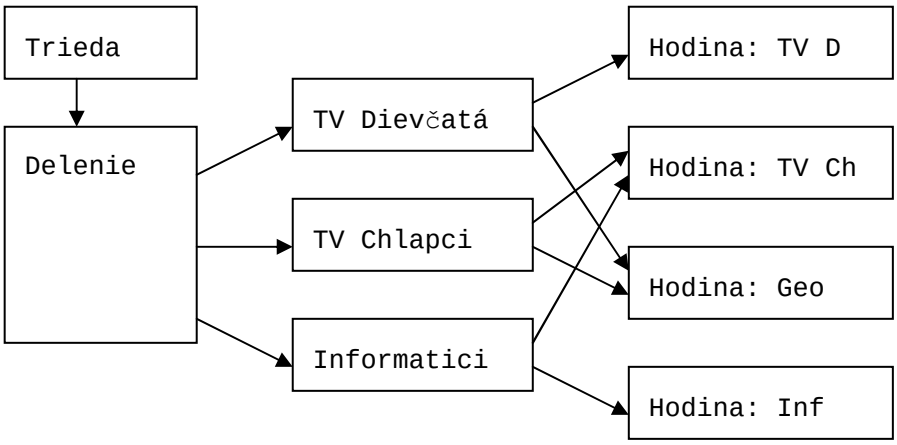
Motivačný príklad

Niekedy pri zlom nastavení delení, môžu vznikať deravé rozvrhy. Napríklad si predstavme, že na telocvik sa trieda delí na chlapcov a dievčatá a nezávisle sa delí ešte na informatikov a zemepisárov.



Aj keď v triede náhodou platí, že nemajú ani jednu informatičku, systém by nedovolil umiestiť na ten istý čas TV_dievčatá a Informatiku. Na takýto jav však musí prísť

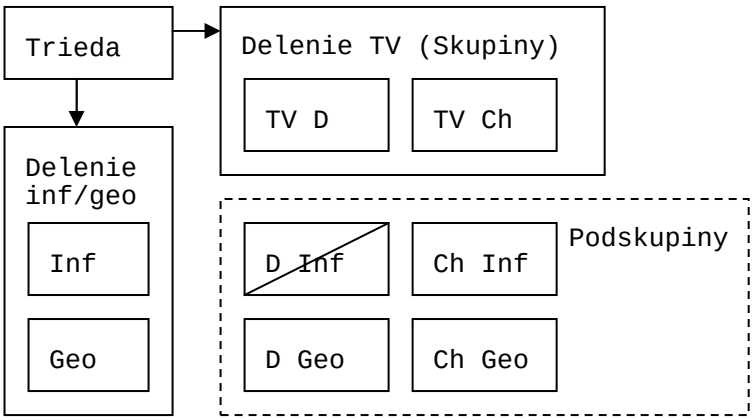
len pozorný operátor a namiesto 2 delení na 4 skupiny definovať jedno delenie na 3 skupiny – zemepisárky, zemepisárov a informatikov.



Riešenie

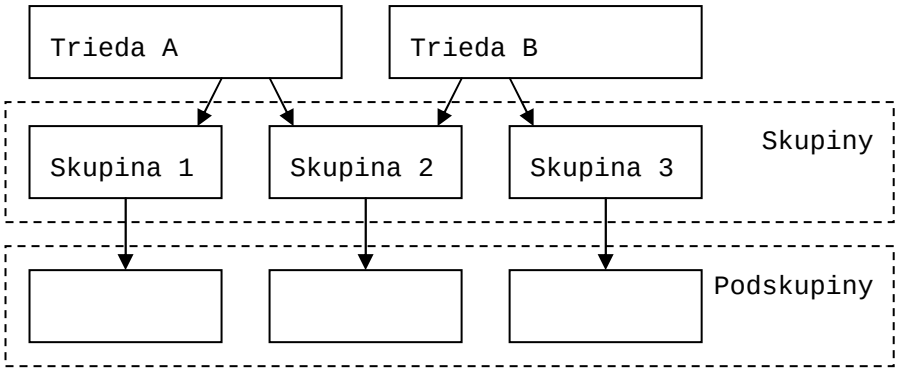
Použijeme zlatú strednú možnosť: Pevne stanovený počet vnorení podskupín. Inšpiráciou bol existujúci program FET, ktorý člení množiny žiakov (student sets) na ročníky, skupiny a podskupiny. (V angličtine years, groups a subgroups.) Pod ročníkom však môžeme chápať tradičnú triedu, pod skupinou ľubovoľnú množinu žiakov (aj z viacerých tried) a pod podskupinou už nedeliteľnú množinu žiakov, so všetkými ostatnými podskupinami disjunktnú. V najhoršom prípade môžeme o podskupine uvažovať ako o jednotlivcovi. Aby sme nemali veľa entít, združíme všetkých žiakov, čo majú všetko vyučovanie spoločné, do jednej podskupiny. Žiak môže patriť len do jednej triedy, avšak do mnohých skupín a napokon zas iba do jednej podkupiny.

Pre zachovanie organizovanosti musíme implementovať nejaký nástroj na správu podskupín. Bude fungovať, ako v motivačnom príklade, no pokúsi sa zostať prehľadný. Pri pridaní ešte jedného delenia napríklad na posilnených a neposilnených z angličtiny, by sa už v hodinách ťažko orientovalo. Používateľ zadá iba jednotlivé kategórie, podľa ktorých sa trieda delí a potom bude môcť v matici vyklikať, ktoré kombinácie neprichádzajú do úvahy:



Do jednotlivých hodín budeme môcť asociovať aj niekoľko množín žiakov, aj keď delenia robíme práve na to, aby vznikla zo žiakov s rovnakou hodinou jedna skupina. Niekedy to však môže ešte sprehľadniť. Napríklad v dvoch triedach nezameraných na matematiku je tak málo žiakov, že sa môžu spojiť a vytovriť jednu skupinu.

Zjavne teda môžeme spájať aj triedy. Navyše, rovnako ako v skupinách z rôznych delení, vieme aj v tomto prípade povedať, ktorá skupina vôbec nezdieľa žiakov s nejakou triedou. Podskupiny sú pred používateľom predvolene skryté.



Na organizáciu sa môžeme pozerať ako na orientovanž graf. Ak majú nejaké dve množiny žiakov neprázdny prienik dosažiteľných podkupín, najskôr zdieľajú žiakov a nesmú byť umiestnené v rovnakom čase na rôzne miesta.

Využitie programu FET timetabler

Program FET je na najbohatší solver rozvrhov čo do druhov obmedzení, aký som doteraz osobne videl. Cieľom tejto práce nie je ho obohatiť. Nanajvyš spríjemniť prácu a korigovanie jeho výsledkov. Má množstvo entít, s ktorými dokáže pracovať:

- building, room
- day, time slot
- student set, teacher, subject
- activity, subactivity – hodina s viacerými výskytmi v rozvrhu (rôznej dĺžky)
- activity tag – ľubovoľná vlastnosť, ktorú vieme pridať hodine. Vystupuje potom v obmedzeniach. (Príklad: „Laboratórne cvičenie“, „vyžaduje prjektor“ a i.)

A množsvto obmedzení, ktoré dokáže nepĺňať:

- maximálny počet okien pre učiteľa/množinu žiakov
- maximálny počet zmien budovy
- rozostup konkrétnych hodín v dňoch
- množina žiakov musí končiť jednou z vymenovaných hodín
- hodina / activity tag / predmet má množinu preferovaných miestností
- ak sú dve hodiny v jeden deň, musia nasledovať
- min/max počet hodín denne/týždenne pre učiteľa/množinu žiakov
- ...

Export do programu FET

FET používa súbory vo formáte XML. Syntax je zjavná z príkladu, najprv vyplníme základné údaje ako názvy jednotlivých časových blokov, kam bude ukladať hodiny (activity), názvy dní, zoznam učiteľov, predmetov, budov, miestností a podobne...

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE FET>
<FET version="5.9.3">

<Hours_List>
	<Number>3</Number>
	<Name>08:10</Name>
	<Name>09:00</Name>
	<Name>9:50</Name>
</Hours_List>

<Days_List>
	<Number>5</Number>
	<Name>Pondelok</Name>
	<Name>Utorok</Name>
	<Name>Streda</Name>
	<Name>Štvrtok</Name>
	<Name>Piatok</Name>
</Days_List>


<Teachers_List>
<Teacher>
	<Name>p. Profesor</Name>
</Teacher>
</Teachers_List>

<Subjects_List>
<Subject>
	<Name>NJ</Name>
</Subject>
<Subject>
	<Name>AJ</Name>
</Subject>
</Subjects_List>

<Buildings_List>
<Building>
	<Name>GJH</Name>
</Building>
</Buildings_List>

<Rooms_List>
<Room>
	<Name>Školské výpočtové laboratórium</Name>
	<Building>GJH</Building>
	<Capacity>15000</Capacity>
</Room>
<Room>
	<Name>Telocvičňa</Name>
	<Building>GJH</Building>
	<Capacity>15000</Capacity>
</Room>
</Rooms_List>
</FET>
```

Každá hodina môže mať niekoľko výskytov. Aby tvorili skupinu, každá má referenciu na prvú zo skupiny. Čísla ID môžeme zvoliť identicky s našimi. FET nepotrebuje, aby išli po sebe.

```
<Activities_List>
<Activity>
	<Teacher>p. Profesor</Teacher>
	<Subject>AJ</Subject>
	<Duration>1</Duration>
	<Total_Duration>2</Total_Duration>
	<Id>1</Id>
	<Activity_Group_Id>1</Activity_Group_Id>
	<Active>true</Active>
	<Students>sexta B AJ pro</Students>
</Activity>
<Activity>
	<Teacher>p. Profesor </Teacher>
	<Subject>AJ</Subject>
	<Duration>1</Duration>
	<Total_Duration>2</Total_Duration>
	<Id>2</Id>
	<Activity_Group_Id>1</Activity_Group_Id>
	<Active>true</Active>
	<Students>sexta B AJ pro</Students>
</Activity>
</Activities_List>
```

FET rozlišuje skupiny pri načítaní podľa ich názvu. Náš program môže vygenerovať pre skupiny škaredé názvy. Preto používateľovi dovolíme si ich prostredníctvom FETu zmeniť. Musí nám však ponechať nedoktnuté naše vlastné ID čísla, ktoré vystupujú v zátvorkách.

```
<Students_List>
<Year>
<Name>sexta B (1)</Name>
<Number_of_Students>36</Number_of_Students>
	<Group>
	<Name>sexta B AJ zac (2)</Name>
		<Subgroup>
		<Name>sexta B AJ zac NJ zac (6)</Name>
		</Subgroup>
		<Subgroup>
		<Name>sexta B AJ zac NJ pro (7)</Name>
		</Subgroup>
	</Group>
	<Group>
	<Name>sexta B AJ pro (3)</Name>
		<Subgroup>
		<Name>sexta B AJ pro NJ zac (8)</Name>
		</Subgroup>
		<Subgroup>
		<Name>sexta B AJ pro NJ pro (9)</Name>
		</Subgroup>
	</Group>
	<Group>
	<Name>sexta B NJ zac (4)</Name>
		<Subgroup>
		<Name>sexta B AJ zac NJ zac (6)</Name>
		</Subgroup>
		<Subgroup>
		<Name>sexta B AJ pro NJ zac (8)</Name>
		</Subgroup>
	</Group>
```

```
        <Group>
        <Name>sexta B NJ pro (5)</Name>
                <Subgroup>
                <Name>sexta B AJ zac NJ pro (7)</Name>
                </Subgroup>
                <Subgroup>
                <Name>sexta B AJ pro NJ pro (9)</Name>
                </Subgroup>
        </Group>
</Year>
</Students_List>
```

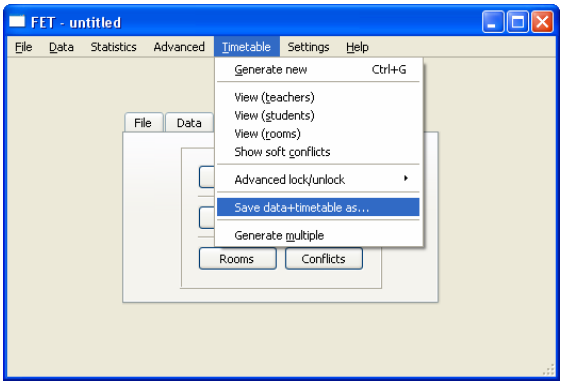
Hodiny, ktoré chceme, aby FET umiestňoval označíme `active`. Hodiny, ktoré máme už v rozvrhu umiestnené, a nechceme, aby ich FET hýbal necháme `active` (aby ich rešpektoval), ale pridáme priestorové a časové obmedzenia. Tag `Permanently_Locked` hovorí, že obmedzenie sa môže v GUI ľahko vypnúť. FET však (vo verzií 5.13.1) nedisponuje žiadnym pohodlným GUI na presúvanie naplánovaných hodín. Jediný spôsob, ako hodiny úmyselne presúvať, je napísanie takýchto obmedzení.

```
<ConstraintActivityPreferredStartingTime>
        <Weight_Percentage>100</Weight_Percentage>
        <Activity_Id>1</Activity_Id>
        <Preferred_Day>Pondelok</Preferred_Day>
        <Preferred_Hour>9:00</Preferred_Hour>
        <Permanently_Locked>false</Permanently_Locked>
</ConstraintActivityPreferredStartingTime>
```

```
<ConstraintActivityPreferredRoom>
        <Weight_Percentage>100</Weight_Percentage>
        <Activity_Id>1</Activity_Id>
        <Room>Školské výpočtové laboratórium</Room>
        <Permanently_Locked>false</Permanently_Locked>
</ConstraintActivityPreferredRoom>
```

Import z programu FET

Program dokáže uložiť výsledný rozvrh takým spôsobom, ako ho ukladáme my. Pre každú umiestnenú hodinu nastaví obmedzenia na uzamknutie v čase aj priestore.



Z takéhoto súboru prečítame iba tieto uzamknutia. Všímame si pri tom iba naše pôvodne vyexportovane IDčka. Ak natrafíme na neznáme, alebo nám bude nejaké chýbať

Suplovanie

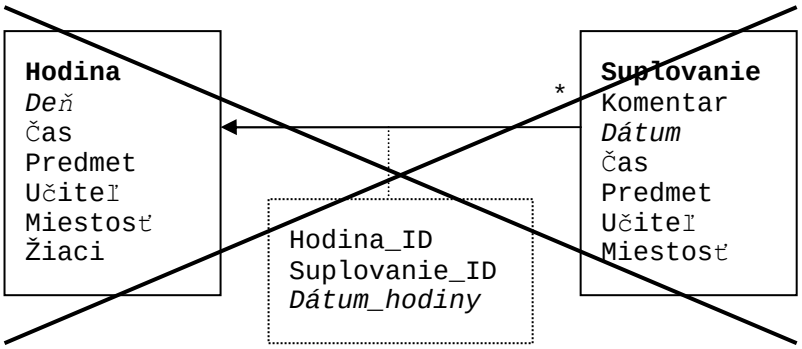
V časti suplovania môže učiteľ nahlásiť svoju neprítomnosť. Zobrazí sa pri tom jednoduchý kalendár s jeho hodinami ako zaškrtávacími políčkami. Môže tak teda spraviť dlhodobo dopredu a na nič nezabudnúť.

Rovnako môže všeobecne zaškrtnúť hodiny v týždni, počas ktorých je ochotný suplovať (napríklad má nevyužité okno). Pri tvorbe suplovania administrátor vidí oddelene skupiny učiteľov, ktorí sú voľní, obsadení alebo chýbajúci. Zoraďujú sa tak, aby navrchu boli tí, čo učia predmet, ktorý treba suplovať.

Štruktúra dát

Treba si uvedomiť, že položka suplovania môže byť rovnako zložitá ako položka hodiny. Líši sa len tým, že namiesto dňa má vyplnený konkrétny dátum a vystupuje v relácií s hodinami, ktoré zastupuje.

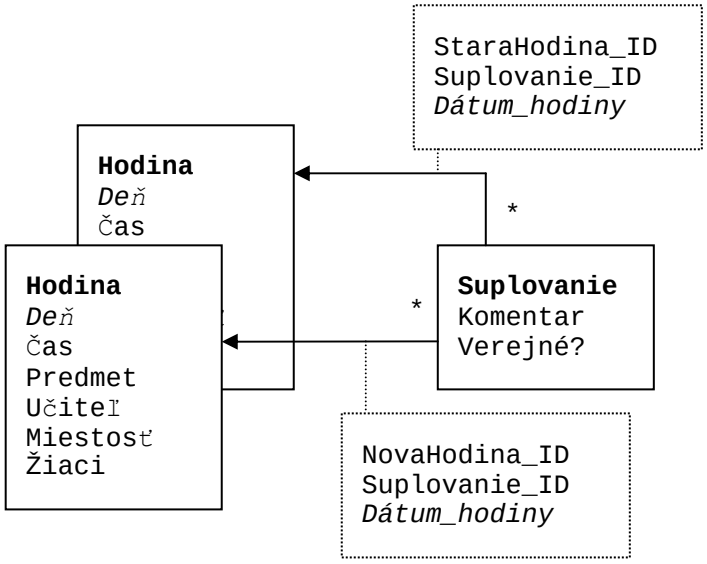
Keď sa (z ľubovoľného dôvodu) supluje hodina, môže sa zmeniť čokoľvek: čas, učiteľ alebo miestnosť... Niekedy aj samotný predmet a dokonca aj deň. Navyše hodina sa môže aj spájať, takže si musíme zvlášť pamätať reláciu medzi suplovanými hodinami a suplovaním. Málokedy nastanú všetky prípady naraz, najčastejšie sa mení iba učiteľ, alebo hodina odpadá, ale všetky tieto zmeny systém umožňuje.



Za povšimnutie tiež stojí špeciálny prípad, keď sa delená hodina zmení na inú delenú hodinu, takže ju nemôžeme reprezentovať ako dve suplovania – každé pre jednu skupinu. Triviálna implementácia načrtnutá vyššie by zlyhala.

Príklad: Majme delenú hodinu angličtiny. Ak ju treba suplovať dá sa každej skupine zvoliť samostatné suplovanie (lebo vystupujú ako samostatné hodiny v rovnakom čase). Dajú sa aj spojiť a zmeniť na matematiku s iným učiteľom. Problém však nastane, keď ju budeme chcieť zmeniť na delenú hodinu nemčiny (s úplne iným delením na skupiny). Ani jedna táto skupina sa nebude dať umiestniť v tom istom čase, lebo niektorí žiaci by vtedy mali byť na angličtine.

Problém treba riešiť radikálne, ako nový rozvrh: suplovanie musí z rozvrhu naozaj odstrániť niektoré hodiny a následne pridať nové. Teraz uplatníme myšlienku z delenia tried na skupiny, že pôvodné aj suplované hodiny budeme vypĺňať do tej istej tabuľky.



Teraz naozaj môžeme zasuplovať viacero hodín viacerými hodinami tak, ako by sme očakávali. Bude nutná malá úprava algorimu vypisujúceho všeobecné rozvrhy bez suplovania: bude musieť ignorovať nové hodiny. Omnoho ľahšie sa však teraz implementujú funkcie ako: vypísanie všeobecného rozvrhu s upozornením na hodiny vystupujúce v suplovaní a vypísanie personalizovaného rozvrhu pre konkrétnu podskupinu.

Suplovanie môžeme vymazať, až keď sú všetky dátumy už minulosťou.

S takýmto modelom by sme mohli skombinovať aj viacej suplovaní do jedného, lebo jednoducho niektoré konkrétne vyučovania odstráni a isté zase pridá. Kľudne aj celý deň, alebo úplne všetko suplovanie, ktoré v syséme máme. Rôzne suplovania však udržujeme kôli prehľadnosti, neobťažovaniu učiteľov mailovými notifikáciami, ktoré sa ich netýkajú a jednoduchému opätovnému zrušeniu.

Nahlásenie neprítomnosti

Ak učiteľ nahlási neprítomnosť najprv sa skontroluje, či v dané hodiny učí a či absencia už nie je evidovaná. (Rovnako, ako pri vypĺňaní známok, môže učiteľ absencie meniť a vtedy nevieme rozlíšiť, či je to nová požiadavka, zmena, alebo jednoducho refresh) Vytvorí sa záznam v tabuľke suplovania, so zatiaľ nevyplnenou novou hodinou. Prípustné akcie sú textový komentár (napr. „Obed“), zmena hodiny (zmení sa čas, miesto, učiteľ, predmet) a spojenie hodín. Rozdiel medzi spojením hodín a zmenou hodiny je iba v mohutnosti relácie medzi daným suplovaním a suplovanými hodinami. Textový komentár napokon môže obsahuje každé suplovanie.

Administrátorovi sa pošle o udalosti mail.

Zmena v suplovaní

Vždy, keď sa ide udiať nejaká zmena v suplovaní, urobí sa zoznam doknutých učiteľov pred zmenou a spojí sa so zoznamom po zmene. Týmto učiteľom sa následne pošle mail s aktuálnym stavom (nanovo sa vyhľadá v databáze). Treba myslieť aj na dátum, aby sme nainformovali aj učiteľov, čo dané suplovanie už odučili. Učitelia z hodiny, ktorá

stratila väzbu so suplovaním (prestala byť spojenou hodinou), dostanú mail oznamujúci, že ich hodina nie je suplovaná a prebehne normálne.

Žiakom sa žiadne maily pri akcií neposielajú. Ľahko by zbytočne vznikal chaos. Žiaci vidia online to suplovanie, ktoré je označené ako už verejné.

Admin v zobrazení suplovania má na každom riadku linku edit. Z roletových ponúk môže vybrať dátum (nie deň) a miestnosť. Na hľadanie týchto môže použiť otvorenie rozvrhu miestností do nového okna. Podľa zvoleného dňa a miesnosti dostane ponuku časov krát učiteľov. Nad touto tabuľkou rovnobežne s časovou osou vidí zvýraznené časy, kedy je trieda a miestosť dostupná. Časy, kedy je voľný aj učiteľ sú zvýraznené. Čas v ktorom je voľná aj vybraná miestnosť je mimoriadne výrazný. Kliknutím do tabuľky určí čas a učiteľa. Všetko ešte potvrí stlačním OK.

Komplikovanejšie suplovanie je už len s viacerými suplujúcimi učiteľmi. Na tento problém už nie je prichystaná ďalšia tabuľka s rozvrhmi učiteľov. Aj tak je táto úloha už tak špecifická, že vyžaduje hlbšie zaangažovanie. Stále je však k dispozícií zoznam učiteľov, v ktorom sa dá vybrať aj viac možností. Pri bežnom použití v ňom netreba vybraťnikoho.

Zobrazenie suplovania

Suplovania sa zobrazujú ako riadky tabuľky zoskupené a zoradené podľa dátumu. V riadku sú vľavo vymenované zrušené hodiny a vpravo ich nahrádzujúce. Ak je v riadku odkaz na hodinu z iného dňa, je uvedená aj s dátumom. Ak je tento dátum už starší ako dátum celej skupiny, v ktorej je riadok vypísaný, je preškrtnutá. Riadok je vypísaný v každej dátumovej skupine, s ktorou má aspoň jednu spoločnú hodinu.

Implementácia

Celý systém ZK je napísaný v jazyku PHP. Výsledkom PHP skriptov sú XHTML 1.0 Strict stránky s použitím CSS a JavaScriptu. Ako databáza sa používa MySQL. Ako webserver je možné použiť napríklad Apache. Prostredie je zvolené tak, aby ho nebol problém nainštalovať bez veľkých znalostí zo sveta IT. Existujú implementácie pre systém Windows aj Linux, dokonca existujú hotové balíčky všetkých troch systémov, ktoré sú však určené pre začiatočníkov alebo vývojárov a vyžadujú si aspoň malý zásah do konfigurčných súborov kôli zabezpečeniu, aby mohli bežať verejne dostupne.

Programátorské konvencie

Globálne nastavenia

Globálne nastavenia, ako napríklad heslo do databázy, alebo doba expirácie rodičovského hesla sú uložené v PHP súboroch v adresári „settings“. Žiadny z podadresárov aplikácie by nemal byť pre ňu z bezpečnostných dôvodov zapisovateľný. Jedinú výnimku tvorí adresár „settings/new“, ktorý sa zase za normálnych okolností nebude snažiť čítať, ale do ktorého bude vytvárať nový súbor s nastaveniami „settings.php“ z administrátorského rozhrania. Týmito potom admin na serveri nahradní pracovné nastavenia. Týmto je zabezpečené, aby sa ani pri úniku administrátorovho hesla nedali meniť kľúčové nastavenia.

HTML special chars

Kedy ošetrovať výstup, ktorý sa má ocitnúť na HTML stránke? Nakoľko stringové premenné často využijeme na viacero účelov (na stránku / do databázy) a prípadne v rôznych kódovaniach udržujeme ich v premenných v neošetrenej podobe. Ošetria sa vždy až tesne pred tým, ako sa majú použiť. Špeciálne na začiatku scriptu overíme, či je zapnutá funkcia „magic quotes“, ktorá automaticky ošetrí všetky vstupy prichádzajúce od klienta a oštrenia odstránime.

Príkladom môže byť napríklad celé meno žiaka, ktoré nemá inú funkciu ako byť vypísané na obrazovku. V databáze ho udržujeme v čistom tvare a funkciu htmlspecialchars() použijeme až pri jeho zobrazení. Odpadá tým napríklad problém so skracovaním mena, keby sa nezmestilo do stĺpca tabuľky – netreba špeciálne kontrolovať, či sme pri skrátení porušili nejakú HTML entitu.

```
strrpos($string, '&') > strrpos($string, ';')
```

Jedinú výnimku tvoria funkcie, ktoré zostavujú HTML výstup. Také práveže očakávajú, že ich parametre budú už spracované na rovnaký účel, na ktorý sú určené aj dané funkcie. Dôvod je prostý: takéto funkcie niekedy treba reťaziť. Pseudopríklad:

```
echo '<table>';
echo tr_parity_style(td(htmlspecialchars('content')));
echo '</table>';
```

Zároveň ako pomôcka funguje to, že funkcia HSC sa nevytratí z očí a ak je niekde echo, musí tam byť aj HSC.

Databázové konvencie

- Tabuľka s heslami na autentifikáciu je oddelená od tabuľky užívateľov, aby sa v budúcnosti ľahšie doimplementovali ďalšie možnosti autentifikácie.
- Každá samostatná databázová úloha (ako napríklad „vypíš zoznam žiakov tohoto rodiča“) je implementovaná v samostatnej funkcii. Prioritou je stručnosť a oddelenie od všetkých nepotrebných (globálnych) premenných. Ak je zobrazenie na stránku komplikovanejšie (pridanie formulárov, JavaScriptov...) má ho riešiť volajúca funkcia. Môže využívať iba globálne nastavenia, ako napríklad názvy tabuliek. Toto sa hodí na oddelenie SQL, pre budúce portovanie systému na iný databázový systém (OstgreSQL).

Štýl známok

Uplatnenie štýlu je implementované ako nahradenie všetkých stringových reprezentácií známok z databázy HTML kódom. Pozrime sa na niektoré úskalia, či riešenia, ktoré stoja za zmienku.

Príklad súbru „settings/markstyles.php“ pre naštýlovanie uzavretých známok:

```
$markstyles = array(
  'V1' => '<span class="mark_V1">výborný</span>',
  'V2' => '<span class="mark_V2">chválitebný</span>',
  'V3' => '<span class="mark_V3">dobrý</span>',
  'V4' => '<span class="mark_V4">dostatočný</span>',
  'V5' => '<span class="mark_V5">nedostatočný</span>',
)
```

Súborom „settings/markstyles.css“ potom definujeme štýly podľa ľubovôle. Je zrejmé, že tento súbor môže editovať iba admin (skutočný vlastník súborov alebo správca servera, nie rola „admin“ vrámci systému). Učitelia môžu do poľa so známkami písať aj iné hodnoty ako vymenované, tých sa systém nedotkne. Rozhodne však pred vypísaním musia prejsť funkciou HtmlSpecialChars. Nahradené naštýlované známky zas nesmú. Na správne zobrazenie známok spravímte teda tieto kroky:

1. Odstrániť prebytočné medzery spomedzi známok
2. HtmlSpecialChars na string so známkami
3. HtmlSpecialChars na kľúče poľa $markstyles (aby mohli byť definované aj menovky so špeciálnym HTML znakom v názve)
4. Usporiadať pole $markstyles podľa dĺžky kľúčov klesajúco
5. Nahradiť v známkach všetky výskyty nejakej menovky jej zobrazením

Zjavne nie je vhodné, aby menovky obsahovali medzery (aj keď by sa tým mohli dosiahnuť zaujímavé efekty). Vďaka usporiadaniu kľúčov podľa dĺžky sa môžu menovky navzájom obsahovať. Prednosť dostane dlhšia.

Ukladanie a nahrávanie údajov

Jednotlivé tabukľky v rámci systému boli navrhované tak, aby boli kompatibilné s formátom súboru .fet. Nakoľko sa jedná o XML, dala by sa v budúcnosti implementovať možnosť ukladania a nahrávania aspoň častí súvisiacich s rozvrhom tak, aby ich dokázal nahrať FET,

Preklad

Preklad nie je dôležitý len kôli možnosti komunikovať v inom jazyku, ale zároveň robí aplikáciu použiteľnú na viac učelov. Stačí napríklad preložiť texty, čo zobrazuje a už sa dá použiť napríklad na plánovanie skautských výletov, alebo organizáciu vedúcich smeny. Aj niektoré školy sa môžu líšiť v názvosloví. Obzvlášť vysoké a stredné. Na jedných sa môže hovoriť prednášky, na niektorých hodiny, na iných kurzy...

Na preklad PHP aplikácií sú v podstate 3 možnosti: Veľké stringové pole, hlášky uložené do databázy a gettext().

Pri veľkom poli sa obyčajne jedná o samostatné súbory, či dokonca celé adresárové štruktúry PHP súborov, ktoré v sebe majú iba definíciu poľa stringov, do ktorého kľúčom sú (skrátené) hlášky v pôvodnom jazyku. Výhoda tohoto spôsobu jednoduchosť pre vývojára alebo prekladateľa. Stačí hlášku preložiť a uložiť PHP súbor a hneď sa zmena prejaví. Zálohovanie starých prekladov je tiež jednoduché a na preklad stačí iba textový editor (musí podporovať vyžadovaný encoding). Nevýhodou je, že pri každom jednom behu skriptu, ktorý daný súbor vyžaduje na preklad, sa musí celý súbor spracovať znovu (PHP udržuje indexy do polí v stromoch), pričom sa skoro nikdy nepoužijú všetky spracované hlášky.

Ďalšou možnosťou je ukladanie prekladov do databázy. Tento spôsob je vhodný pre CMS systémy, ktoré majú svoje obsahovať články vo viacerých jazykoch. Pre vývojára je tiež náročnejší, lebo musí aplikácií prirobiť prekladací interface. Výhodou je, že v databáze sa vyhľadajú len tie hlášky, ktoré sa naozaj musia zobraziť a databáza to vie rýchlejšie než lineárne prejdenie a spracovanie PHP súboru.

Treťou možnosťou je GNU gettext. Jedná sa o rozšírenú platformu na prekladanie software-u. Prekladateľ pripraví šeciálne formátovaný textový dokument s prekladom a následne ho skompiluje do binárnej podoby nástrojom gettextu. V tejto podobe sa v ňom rýchlo vyhľadáva. Knižnice gettextu existujú pre množstvo programovacích jazykov, aj pre PHP. Výhodou je rýchle vyhľadávanie prekladov aj bez použitia databázy a možnosť prepínať hláškové domény. Gettext myslí aj na rôzne znejúce hlášky v jednotnom a množnom čísle a správne skloňovanie s číslovkami. Medzi nevýhody môžeme zaradiť obtiažnosť opravy prekladu v prípade skompilovaného jazykového súboru a nutnosť mať PHP interpreter s podporou pre gettext.

Pridávanie zásuvných modulov

Zásuvné moduly sa nachádzajú v podadresári „plugins“ ako ďalšie podadresáre, ktoré musia obsahovať súbor „setup.php“. Tento súbor nemá pridávať žiadne globálne premenné, iba zaregistrovať mená k jednotlivým triedam, prípadne modifikovať niektoré menu.

V ostatných ohľadoch fungujú rovnako ako triedy zobraziteľných komponentov, ktoré sú bežnou súčasťou aplikácie.

Bibliografia

[1] editors Mehdi Achour et al. Php manual. http://www.php.net/manual/en/, 2010.

[2] W3C. Extensible markup language (xml) 1.0 (fourth edition). http://www.w3.org/TR/REC-xml, 2006.

[3] Liviu Lalescu, Volker Dirr. FET documentation http://lalescu.ro/liviu/fet/doc/

[4] The Apache Software Foundation. Apache HTTP Server Documentation. http://httpd.apache.org/docs/2.2/, 2009

[5] Pratap Lakshman, Allen Wirfs-Brock. ECMAScript ECMA-262 specification, 2009

[6] W. Jason Gilmore. Velkľa kniha PHP5 a MySQL. Zoner Press, 2004.

[7] editors Mehdi Achour et al. Php manual. http://www.php.net/manual/en/, 2010.