TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

BAKALÁŘSKÁ PRÁCE

Liberec 2013 **Pavel Bucháček**

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Matrika pro mateřské školy

Registry of nursery school

Bakalářská práce

Autor: **Pavel Bucháček**
Vedoucí práce: Ing. Petr Kretschmer
Konzultant: Ing. Vojtěch Wrnata

V Liberci 17. 5. 2013

Fakulta mechatroniky, informatiky a mezioborových studií

Zadání bakalářské práce

Příjmení a jméno studenta	Pavel Bucháček (M10000132)
Zkratka pracoviště	NTI
Datum zadání BP	18. 10. 2012
Plánované datum odevzdání	17. 5. 2013
Rozsah grafických prací	dle potřeby dokumentace
Rozsah průvodní zprávy	cca 45 stran
Název BP (česky)	Matrika pro mateřské školy
Název BP (anglicky)	Registry of nursery school

Zásady pro vypracování BP:

1. Analýza systému, požadavky na něj, vytváření výstupních sestav
2. Návrh databázové struktury
3. Realizace vlastního systému (pro více uživatelů)
4. Ověření funkčnosti
5. Vytvoření instalační distribuce a stručného návodu k použití

Seznam odborné literatury:

[1] Gilmor, J. W., Velká kniha PHP a MySQL 5 - kompendium znalostí pro začátečníky i profesionály, Zoner Press, 2007

[2] Kosek, J., PHP - tvorba interaktivních internetových aplikací, Grada Publishing, 1999, ISBN 80-7169-373-1

Vedoucí BP	Ing. Petr Kretschmer
Konzultant BP	Ing. Vojtěch Wrnata

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum: 17. 5. 2013

Podpis:

Poděkování

Rád bych tímto poděkoval panu Ing. Petru Kretschmerovi za všechny cenné rady, které mi poskytl. Děkuji za podmětné připomínky, nápady a čas, který mi věnoval. Dále bych chtěl poděkovat paní doktorce Kamile Podrápské za vstřícný přístup, nápady a připomínky, které mi v průběhu tvorby aplikace poskytla.

Abstrakt

Cílem bakalářské práce je vytvořit matriku pro potřeby mateřské školy v podobě webové aplikace. Aplikace je navržena tak, aby byla do budoucna snadno rozšiřitelná, dbala důrazu na bezpečnost a bylo ji možné lokalizovat do libovolného jazyka. Systém je postaven na vlastním PHP frameworku, jehož základ byl položen v rámci bakalářského projektu. V aplikaci využívám objektové programování a návrhové vzory. Z důvodu snadné rozšiřitelnosti a udržovatelnosti je aplikace postavena na architektuře MVC (model view controller).

Framework abstrahuje základní prostředky využívané ve webových aplikacích a jeho hlavními výhodami jsou jednoduchost a větší výkonnost ve srovnání s velkými PHP frameworky. Framework také obsahuje nástroje pro programové generování databáze, export a import dat.

Z důvodu uchovávání osobních údajů je nutné klást důraz na bezpečnost dat ukládaných v systému. Framework řeší známé bezpečnostní slabiny webových aplikací a při nasazení aplikace do praxe je použit šifrovaný protokol HTTPS. Veškerá uživatelská hesla jsou uchovávána v podobě salted hash, což znemožní použití duhových tabulek pro získání původního hesla.

Aplikace je víceuživatelská, uživatelé jsou členěny do skupin, kterým lze přidělovat práva na jednotlivé akce. Systém je postaven modulárně, do budoucna je tedy možné jej v případě potřeby jednoduše rozšířit.

V matrice se uchovávají všechna potřebná data o dětech navštěvujících školku, jejich rodičích a učitelích. Děti je možné řadit do školních roků a tříd. Matrika byla vyvíjena ve spolupráci s MŠ Klášterní v Liberci, která má také speciální třídy. Aplikace tedy umožňuje vytvářet třídy s kolektivními integracemi dětí, nebo jim přiřazovat individuální integrace. V databázi se udržuje kompletní historie o pobytu dítěte. Je také možné spravovat informace o školních pobytech dětí. Tyto informace je možné tisknout ve formě několika typů tiskových sestav, které jsou nezbytné pro potřeby mateřské školy. Samostatný modul je také věnován správě uživatelů programu, jejich členění do skupin a přidělování práv.

Klíčová slova: matrika, PHP framework, webové aplikace, MVC, objektové programování

Abstract

Subject of this bachelor thesis is creation of web application for register of nursery school. The application is designed to be easily extended in the future. It is focused on security and it is possible to localize it in any language. The system is based on my own PHP framework which was created in bachelor project. Application uses object oriented programming, and design patterns. The application is based on model view controller architecture.

The purpose of framework is abstraction of basic tools used in web applications and its biggest advantages are the simple usage and bigger performance in comparison with other big PHP frameworks. Framework also contains tools for generating database, export and import of data.

Because of storing sensitive information it is very important to take care of security. The framework solves all well-known security risks of web applications and in production mode will be used protocol HTTPS. All user passwords are stored in database as salted hash, so attacker cannot use rainbow table to get original password.

System is multiuser application and users are structured in groups. Privileges to individual actions are assigned to groups. The system is modular, so it is easy to extend it in the future.

All necessary information about children, their parents and teachers are stored in the registry. You can assign children into school year and nursery class. The registry of nursery class was developed in co-operation with nursery school Klášterní in Liberec with special classes. Application supports creation of classes with collective integration of children and assigns individual integration to children. Full history of children stay is stored in the database. You can also store information about school trips. All these information can be exported as printout, which is necessary for all nursery schools. There is also a module for managing users, assigning users to groups and assigning privileges to groups.

Key words: registry of nursery school, PHP framework, web application, MVC, object oriented programming

Obsah

Seznam obrázků

Seznam symbolů, zkratek a termínů

Ajax (Asynchronous JavaScript) – Označení pro technologii, pomocí které lze asynchronně komunikovat mezi klientem a serverem. Je tedy možné provést výměnu dat bez znovunačtení webové stránky.

Apache – Multiplatformní softwarový webový server s otevřeným zdrojovým kódem.

Autentizace – Bezpečnostní opatření, při kterém se ověřuje identita subjektu.

Autorizace – Proces povolení přístupu k určité akci.

CSS (Cascading Style Sheets) – Kaskádové styly, jazyk pro popis způsobu zobrazení webových stránek, odděluje vzhled HTML dokumentu od jeho obsahu a struktury.

DOM (Document Object Model) – Objektový model dokumentu, jedná se o objektovou reprezentaci HTML dokumentu, pomocí které můžeme přistupovat k jednotlivým prvkům a pracovat s nimi.

FCP – Zkratka z anglických slov front controller pattern, jedná se o návrhový vzor používaný u webových aplikací. Uživatel systému přistupuje vždy jen k jednomu souboru (nazývaný bootstrap), ten se poté stará o vyřizování všech požadavků.

Framework – Softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů.

GNU – Projekt, který je zaměřen na svobodný software. Původním cílem bylo vyvinout vlastní operační systém. Pod hlavičkou GNU vzniklo několik zajímavých projektů a licencí (např.: GPL a GFDL).

GPL (GNU General Public License) – Licence, která vyžaduje, aby byla odvozená díla dostupná pod toutéž licencí.

Hashovací funkce – Jedná se o matematickou funkci, která pro řetězec o libovolné délce vrátí unikátní řetězec konstantní délky. Používá se například pro uchování hesel v databázi.

Hibernate – Framework pro objektově relační mapování dat napsaný v jazyce Java.

HTML – Značkovací jazyk, ve kterém se vytvářejí webové stránky.

JavaScript – multiplatformní, interpretovaný skriptovací jazyk, často používaný pro interaktivní prvky ve webových aplikacích.

Jmenné prostory – Dovolují oddělit jednotlivé části programu tak, že i když obsahují stejnojmenné identifikátory, nebudou vzájemně kolidovat.

mod_rewrite – Modul webového serveru Apache, zpracovává přicházející url požadavky a na základě konfiguračního souboru (.htaccess) řídí přesměrování a nastavuje stavové hlavičky.

MVC – Softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní.

Návrhový vzor – Představuje obecné řešení problému, které se využívá při návrhu programů. Jedná se o popis řešení problému nebo šablonu, která může být použita v různých situacích.

Návrhový vzor Registry – Návrhový vzor, který slouží pro distribuci sdílených proměnných napříč aplikací.

ORM – Objektově relační mapování, automatická konverze dat získanými z relační databáze na objekty a naopak.

PHP – Skriptovací programovací jazyk, který je určený především pro programování dynamických internetových stránek a webových aplikací.

Rainbow table – Duhová tabulka, jedná se o kolekci heslových frází, u kterých je dopředu spočítán jejich hash. Útočník je používá pro získání původních hesel z ukradené databáze.

Singleton – Návrhový vzor, který se používá při řešení problému, kdy je potřeba, aby v celém programu existovala pouze jedna instance třídy.

SQL – Jedná se o dotazovací jazyk, který se používá v relačních databázích pro práci s daty.

Šablonovací systém – Pracuje se šablonami obsahujícími značky, se kterými šablonovací systém manipuluje, vyhodnocuje je a nahrazuje za konkrétní data. Výstupem šablonovacího systému je výstupní dokument – například webová stránka.

Trigger (spoušť) – Jedná se o automatizovanou činnost, která se vyvolá před, nebo po definované události nad tabulkou.

1 Úvod

Pro chod mateřské školy je nezbytně nutné udržovat data o jednotlivých dětech, jejich rodičích a třídách, které navštěvují. Mezi těmito daty je potřeba se jednoduše orientovat a po skončení školního roku vykázat kolik dětí školku navštěvovalo. Ještě složitější situace je u školek se speciálními třídami, které musí evidovat informace o typech integrace dítěte a jejich vývoji. Veškeré tyto informace je potřeba uchovávat také v papírové podobě.

Je až zarážející, že v současnosti neexistuje žádné softwarové řešení, které by řešilo zmíněnou problematiku. Existují programy na správu žáků pro základní, střední i vysoké školy, ale na mateřské školy jako by se zapomnělo. Na základě požadavku ředitelky MŠ Klášterní v Liberci paní doktorky Kamily Podrápské vznikl nápad na tvorbu tohoto systému.

Doposud musela všechny zmíněné informace udržovat v podobě tabulek pomocí programu Excel. Orientace v datech byla velmi složitá a časově náročná, o tiskových výstupech ani nemluvě. Program pro matriku zaměstnancům mateřské školy usnadní práci, ušetří spoustu času a starostí.

Z důvodu snadné dostupnosti a správy programu jsem se rozhodl pro formu webové aplikace. Tento typ aplikace je možné využívat napříč všemi zařízeními a platformami. Odstraní se také problémy s distribucí případných aktualizací programu.

Aplikaci jsem postavil v jazyce PHP, ve kterém mám největší zkušenosti a jako úložiště dat jsem zvolil relační databázi MySQL. Při tvorbě programu jsem použil PHP framework, jehož základy jsem položil v bakalářském projektu. Pro potřeby matriky musel být vhodně doplněn o potřebné části. Aby byla aplikace uživatelsky přívětivá, využívám některých doplňků z balíku jQuery UI a pro samotný vzhled aplikace používám kaskádové styly Twitter Bootstrap.

2 Použité technologie

2.1 PHP

Aplikace je napsaná v jazyce PHP – jedná se o serverový skriptovací jazyk, který se používá zejména pro tvorbu webových stránek a aplikací. PHP je dynamicky typovaný jazyk a s výhodou v něm lze využívat dynamických polí. Od verze 5 nabízí pokročilé možnosti objektového programování [1] a použití jmenných prostorů.

2.2 MySQL

Pro ukládání dat využívám relačního databázového systému MySQL. Jedná se o multiplatformní databázi, která využívá jazyk SQL. V závislosti na jejím využití je dostupná pod dvěma typy licence: pod licencí GPL a pod komerční licencí. V poslední době podporuje triggery [2], uložené procedury a pohledy. Je velmi rozšířená a spolu s jazykem PHP a serverem Apache tvoří základ většiny dnešních webových aplikací.

2.3 jQuery

Jedná se o JavaScriptovou knihovnu [3], která se řídí mottem „Write less, do more.“, v překladu tedy „Napiš méně, udělej více.“. Usnadňuje výběr elementů v DOM, snadno jim přiděluje události, jednoduše manipuluje s CSS styly, nabízí celou řadu efektů a animací. Velmi také usnadňuje asynchronní komunikaci pomocí Ajaxu. V tomto frameworku je napsána celá řada utilit, které usnadňují a zpříjemňují práci s webovými aplikacemi.

2.4 Twitter Bootstrap

Bootstrap [4] je sada CSS stylů, HTML komponent a volitelných JavaScriptových rozšíření, které usnadňují sestavování grafického rozhraní u webových aplikací. Dal by se označit za design framework. Kromě samotného usnadnění práce je další výhodou optimalizace pro všechny internetové prohlížeče a rozlišení obrazovky.

2.5 jQuery UI

JavaScriptový framework, který stejně jako samotné jQuery spadá do projektu The jQuery Project, vznikl za účelem usnadnit vývojářům práci s použitím pokročilých efektů a ovládacích prvků ve webových aplikacích [5]. Framework má čtyři základní části:

- **Interactions –** obsahuje metody pro pokročilou interakci mezi aplikací a uživatelem,
- **Widgets** – pokročilé prvky grafického prostředí (např.: progressbar, vyskakovací okna, datepicker),
- **Utilities** – pozicování widgetů,
- **Effect**s **–** grafické efekty a animace.

2.5.1 Timepicker

Widget sloužící pro výběr data ze zobrazeného kalendáře. Je možné jej libovolně nastylovat a upravit pro konkrétní potřeby aplikace. Pro potřeby matriky muselo dojít k přeložení jeho popisků do češtiny a zobrazení selectů pro výběr měsíce a roku.

```
$(function() {
  $(".datepicker").datepicker({
     dateFormat: "yy-mm-dd",
     showButtonPanel: true,
     changeMonth: true,
     changeYear: true
  });
});
```

2.5.2 Multiselect

V několika případech je v aplikaci nutné vybírat několik hodnot ze seznamu. V jazyce HTML k tomu slouží prvek `select`, kterému se nastaví atribut `multiple`. Tento ovládací prvek je však uživatelsky velmi nepřívětivý a neintuitivní.

Tyto nevýhody řeší plugin Multiselect. U každé položky je zobrazeno zaškrtávací tlačítko, po zabalení seznamu se mohou vypsat vybrané hodnoty, nebo jejich počet. Prvek jsem ještě doplnil o ovládací prvky pro vybrání všech hodnot, zrušení výběru a filtračního pole – uživatel do něj může napsat hledaný řetězec a seznam hodnot se automaticky vyfiltruje.

```
$(document).ready(function() {
   $(".multiselect").multiselect({
      selectedList: 10,
      checkAllText: "Vybrat vše",
      uncheckAllText: "Odznačit vše",
      noneSelectedText: "Nic nevybráno",
      selectedText: "Vybráno # položek",
      minWidth: 450,
      height: 250
   }).multiselectfilter({
      label: 'Filtr:',
      width: 130,
      placeholder: 'Zadejte hledaný výraz'
   });
};
```

2.5.3 Tablesorter

Po výpisu dat do tabulky uživatel většinou očekává možnost si data setřídit podle některého atributu. Častým řešením je odeslání GET požadavku na server s předanými parametry. Server poté z databáze načte data seřazená podle zvoleného atributu a skript uživateli znovu vykreslí stránku.

Aby nedocházelo ke zbytečným požadavkům na aplikační na databázový server, je lepší celou operaci provést na straně klienta pomocí JavaScriptu.

Plugin Tablesorter řeší právě tento problém: umožňuje řadit podle libovolného počtu parametrů jak vzestupně, tak sestupně. Jediný problém nastává při třídění textových řetězců, kdy plugin špatně vyhodnocuje znaky české abecedy. Naštěstí na tuto možnost autoři mysleli, lze tedy dopsat vlastní parser, s jehož využitím se budou data řadit.

Parser pro českou abecedu obsahuje pole se všemi znaky české abecedy. Následně prochází postupně všechny znaky vstupního slova. Každému znaku přiřadí číselnou váhu podle pozice písmena v abecedě, čím je písmeno ve slově dále, tím má menší váhu. Takto spočtené váhy se sčítají a konečný součet funkce vrací – ve finále plugin výsledky seřadí právě podle tohoto čísla.

```
$.tablesorter.addParser({
   // set a unique id
   id: 'czechAlphabet',
   is: function(s) {
      // return false so this parser is not auto detected
      return false;
   },
   format: function(s) {
      s = s.toLowerCase();
      var result = 0;
      var alphabet = ["a", "á", ..., "ž"];
      for (var i = 0, len = s.length; i < len; i++) {
         var increment = (i == 0) ?
            alphabet.indexOf(s[i])
            : alphabet.indexOf(s[i]) / Math.pow(100, i);
         result += increment;
      }
      return result;
   },
   // set type, either numeric or text
   type: 'numeric'
});
```

U každé tabulky, která využívá tento plugin, se definuje jaký parser používají jednotlivé atributy a klávesa, která slouží pro výběr více atributů.

```
$("#listOfChildrens").tablesorter({
   sortList: [[1,0], [2,0]],
   headers: {
      0: {sorter: false},
      1 : {sorter: "czechAlphabet"},
   },
   sortMultiSortKey: 'ctrlKey'
});
```

3 Návrh struktury aplikace

Celá aplikace je od začátku navržena tak, aby ji bylo do budoucna možné snadno rozšiřovat. Důraz byl proto dbán na správné členění jmenných prostorů, adresářové struktury a použití návrhových vzorů.

Při tvorbě webových aplikací je potřeba využívat operace, které se opakují u každého programu. Z tohoto důvodu se používají aplikační frameworky, které obsahují základní nástroje pro tvorbu aplikací. Programátor poté neřeší rutinní operace (komunikace s databází, ověření autentizace uživatele, atd.), ale zabývá se pouze business logikou aplikace.

Pro jazyk PHP je k dispozici celá řada frameworků: mezi nejznámější patří Zend Framework, Symfony, nebo české Nette. Tyto frameworky obsahují celou řadu mocných nástrojů, které mohou ulehčit programátorskou práci. Před jeho použitím se však programátor musí nejdříve seznámit s programátorskými technikami daného frameworku. Zde se dostáváme k největší nevýhodě: buďto je jeho používání neintuitivní, nebo narazíme na neaktuálnost dokumentace. Tyto faktory nám ztěžují práci s frameworkem. Pokud pomineme tyto nevýhody, tak některé frameworky obsahují zbytečně mnoho nástrojů, jejich použití je poté náročnější na vytížení serveru.

Z těchto důvodů jsem se rozhodl k vytvoření jednoduchého PHP frameworku, který bude obsahovat pouze základní nástroje, které se v aplikaci skutečně využijí. Vzhledem k malému rozsahu bude jednoduché se v něm zorientovat. Zároveň je napsán tak, aby do něj v případě potřeby bylo jednoduše možné doplnit libovolnou komponentu.

Na tomto frameworku je postaven zbytek aplikace, která z důvodu snadné rozšiřitelnosti využívá návrhový vzor MVC (model view controller).

3.1 Struktura frameworku

Podle výše zmíněných požadavků je celý framework postaven modulárně. Jednotlivé části potřebují komunikovat jak s ostatními komponentami frameworku, tak se zbytkem aplikace. Pro zajištění tohoto požadavku jsem podle vzoru Michaela Peacocka [6] použil návrhový vzor Registry, který implementuje návrhový vzor Singleton. V systému tedy existuje vždy jen jedna instance tohoto objektu, který obsahuje kolekci s referencemi na ostatní části frameworku. Reference registru se poté pomocí konstruktoru předává všem třídám v aplikaci (viz Obrázek 1). Kromě kolekce s referencemi obsahuje registr také pole s nastavením aplikace a metody, které se využívají v různých místech aplikace – nejsou tedy vázány na konkrétní objekt.

3.2 Části frameworku

3.2.1 Modul pro práci s databází

Tento modul abstrahuje veškeré operace mezi aplikací a databázovou vrstvou. Stará se tedy o spojení s databází (modul umožňuje spravovat i více současných spojení), správu transakcí, zasílání všech typů dotazů a získávání jejich výsledků. Aby aplikace nebyla závislá na konkrétním typu databáze, obsahuje framework abstraktní třídu Database, která definuje všechny metody. V aplikaci je využívána MySQL databáze, ve frameworku je tedy implementována třída Mysql, která dědí od třídy Database. Pokud by v budoucnu bylo nutné přejít na jiný typ databáze, stačilo by nadefinovat novou třídu, která by implementovala všechny abstraktní metody.

Před prací s databází je nejdříve nutné vytvořit nové spojení pomocí metody `newConnection`. Takto vytvořené spojení je následně uloženo do kolekce – je tedy možné vytvořit spojení s více databázemi najednou a mezi nimi přepínat pomocí třídní proměnné `activeConnectionIndex`. Libovolný dotaz na databázi lze zadat pomocí metody `insertQuery`. Pro nejčastěji využívané typy dotazů však existují funkce `delete`, `insert`, `update` a `buildSelect`, které programátorovi usnadní práci a hlavně odstraní závislost na konkrétním typu databáze. Takto vytvořené dotazy se ukládají do kolekce `queries`, jejich výsledky se následně uchovávají v poli `results`.

Příklad definování příkazu select poté vypadá následovně:

```
$id = $this->db->getResults(
      $this->db->orderSelectQuery(
         $this->db->buildSelect(
            Array("id"),
            \Bs\Model\Childrens\ChildrenHistory::TABLE_NAME,
            new \Bs\Registry\Objects\Database\WhereCondition(
              \Bs\Model\Childrens\ChildrenHistory::COLUMN_CHILDREN_ID.
              " = ?", $childrenId)
          )
        )
      );
```

3.2.2 Modul pro práci s formuláři

Jednou z nejčastěji používaných komponent ve webových aplikacích jsou různé typy formulářů – právě proto je modul pro práci s nimi nedílnou součástí frameworku. Základ modulu tvoří třída `Forms`, která obsahuje kolekci všech vytvořených formulářů a nabízí možnost vytvoření nového formuláře. V registru je uchovávána reference na objekt této třídy.

Samotný formulář reprezentuje třída `Form`, která kromě základních informací o formuláři (id, css třída, název, metoda odeslání dat, …) obsahuje pole se všemi prvky

formuláře. Ty umožňuje řadit do vizuálně oddělených kategorií a mezi jednotlivé elementy vkládat textové poznámky.

Formulář umí vykreslit buď v čistém html formátu po zavolání metody `_toString`, nebo ho vykreslí s využitím šablonovacího systému (je nutné zadat cestu k šabloně formuláře). Třída obsahuje metody pro přidání všech existujících elementů. Po odeslání takto vytvořeného formuláře dojde k validaci všech prvků – pokud odeslaná data nevyhovují nadefinovaným validačním kritériím, tak dojde k uložení všech chybových zpráv do pole a následnému předvyplnění polí odeslanými hodnotami.

Každý typ prvku je ve frameworku reprezentován vlastní třídou a lze mu nastavit výchozí hodnotu. Všechny prvky dědí od abstraktní třídy `Element`, která obsahuje základní vlastnosti, definuje abstraktní metody pro validaci a definuje konstanty pro substituce v chybových hláškách. V každém formuláři dojde k automatickému vložení prvku typu `HiddenInput`, podle kterého se identifikuje odeslaný formulář. Základní prvky pro textový vstup jsou reprezentovány třídami `TextInput`, `PasswordInput`, `TextArea` (dědí od třídy `Input`), prvky pro výběr dat tvoří třídy `Select`, `RadioInput` a `CheckboxInput`. Tlačítka jsou realizována pomocí tříd `ButtonInput` a `ButtonFileInput`. Hierarchie všech prvků je zachycena v následujícím seznamu:

- Element
    - Input
        - ButtonInput
            - ButtonFileInput
        - HiddenInput
        - CheckboxInput
        - PasswordInput
        - RadioInput
        - TextInput
    - Select
    - Textarea

Pomocí těchto prvků je možné sestavit libovolný formulář, programátor se poté nemusí starat o validaci dat, ani o opětovné načtení dat do formuláře po odeslání. Ukázka vytvoření přihlašovacího formuláře vypadá následovně:

```
$forms = $registry->getObject("forms");
$forms->newForm("loginForm", "loginForm");
$loginForm = $forms->loginForm;
$loginForm->addTextInput("login", "login")
   ->addValidationRule(Bs\Registry\Objects\Forms\Form::VALIDATION_FILLED,
"You have to type %label.");

$loginForm->addPasswordInput("pass", "password")
```

```
   ->addValidationRule(Bs\Registry\Objects\Forms\Form::VALIDATION_FILLED,
"You have to type %label.");

$loginForm->addSubmitButton("submitForm", "Submit");

if ($loginForm->getSubmited()) {
   $values = $loginForm->submit();
   if (is_null($values)) {
      //invalidate values, print error messages
      foreach ($loginForm->getValidateMessages() as $errorMessage) {
         echo $errorMessage;
      }
   } else {
      //data are valid,
      var_dump($values);
   }
}
```

3.2.3 Modul pro správu autentizace

Další častou akcí u webových aplikací je autentizace, čili ověření uživatele pro vstup do systému na základě předaných přihlašovacích údajů (nejčastěji jméno a heslová fráze).

Tento úkon obstarává třída `Authentication`, obsahující veřejnou metodu `checkAuthentication`, která kontroluje, jestli se uživatel snaží přihlásit. Pokud tomu tak je, zkontroluje přihlašovací údaje proti databázi, pokud jsou korektní a uživatel je aktivní, tak uloží jeho login do session proměnné. Před uložením do session se ještě volá funkce `session_regenerate_id`, která slouží jako ochrana před session fixation. Pomocí direktivy `ini_set` je navíc zakázáno předávání session ID pomocí url.

3.2.4 Modul pro směrování požadavků

Vzhledem k tomu, že aplikace používá návrhový vzor FCP (front controller pattern), tak jsou veškeré požadavky pomocí mod_rewrite (modul serveru Apache) přesměrovány na souboru `index.php`. Hodnota původní adresy se předává pomocí parametru metody GET. Toto směrování je zajištěno pomocí takzvaného podstrčení – uživatel vidí v prohlížeči původní adresu, ve skutečnosti ale požadavky vyřizuje jiný soubor.

Další směrování již probíhá v režii směrovače. Ten umožňuje požadavky přesměrovávat jinam s hlavičkou 301 Moved Permanently. Dále nabízí mód podstrkávání, kdy se pod virtuální url adresou zobrazí soubor, který je fyzicky uložen na jiné adrese. Je-li to možné, zobrazí se obsah souboru přímo v prohlížeči (např. obrázek, textový soubor), v opačném případě dojde k jeho stažení. Pokud nenastane žádná ze zmiňovaných situací, dojde k předání řízení třídě `AdminController` (viz kapitola 3.4.4).

3.2.5 Šablonovací systém

Pomocí šablonovacího systému dochází k důkladnému oddělení datové a aplikační logiky od prezentační části aplikace. Ta je realizována skládáním šablonových souborů s příponou tpl pomocí šablonovacího systému. Tyto soubory obsahují běžný HTML kód, místo datových hodnot ale obsahují šablonovacím systémem definované značky. V kontroleru se šabloně předávají proměnné, které jsou doplněny šablonovacím systémem.

Šablonovací systém umí nahrazovat jednotlivé proměnné, vypisovat kolekce dat (iterace přes jednotlivé položky), vypisovat rekurzivní seznam dat, vyhodnocovat podmínky a podporuje také lokalizaci textových řetězců (viz kapitola 3.5.2).

O tyto úkony se stará třída `Template`, které se nejdříve musí předat cesta k souboru se šablonou. Třída obsahuje metody pro vkládání potřebných datových položek a sadu privátních metod starajících se o vyhodnocení šablony a sestavení výstupu.

Ukázka šablonovacího souboru pro výpis seznamu:

```
<tbody>
   <!--start childrensList-->
      <tr class="childrenRow" id="{id}">
         <td>{lastName}</td>
         <td>{firstName}</td>
         <!--if schoolYear !EMPTY ID 14-->
            <td>{schoolYear}</td>
         <!--endif ID 14-->
      </tr>
   <!--end childrensList-->
</tbody>
```

Předávání proměnných v kontroleru:

```
$tags = Array();
foreach ($childrens as $children) {
   $tags[] = Array(
      "firstName" => $children->getFirstName(),
      "lastName" => $children->getLastName(),
      "schoolYear"  =>  is_null($children->getSchoolYear())  ?  null  :
$children->getSchoolYear()->getName(),
   );
}
$template->addBlock("childrensList", $tags);
$template->getContent();
```

3.2.6 Generátor databáze

Před tvorbou aplikace jsem do frameworku ještě doplnil modul starající se o generování struktury databáze. Upustil jsem od přístupu, kdy se databáze vytvoří v externím nástroji (např. MySQL Workbench, phpMyAdmin, apod.) a na takto namodelované databázi se postaví aplikace. Hlavní nevýhodou je obtížná úprava takto vytvořeného programu. V praxi

se velmi často stává, že po navržení databáze a naprogramování aplikace vznikne na straně zákazníka další požadavek obnášející úpravu struktury databáze. Po úpravě databáze může dojít k tomu, že se na některou z provedených změn zapomene v datovém modelu aplikace a vznikne tak špatně dohledatelná chyba.

Rozhodl jsem se proto pro jiné řešení, jako inspirace mi posloužil ORM framework Hibernate, který používám v Java aplikacích. Využívá se v něm anotací – v datovém modelu se třídní položky a třída samotná anotují (např. anotace `@Table`, `@Column`, apod.). Z takto vytvořeného modelu aplikace se následně vygeneruje celá struktura databáze. K úpravám tedy dochází pouze v modelu aplikace a nehrozí rozhození synchronizace mezi aplikací a databází.

V jazyce PHP bohužel není možné využít anotací jako v Javě. Musel jsem tedy vymyslet jiný způsob, jak definovat databázovou strukturu příslušející datovému modelu. Ve jmenném prostoru `DatabaseUtils` jsem vytvořil sadu tříd, které slouží jako datové struktury pro definici databáze.

Každý datový model, který reprezentuje databázovou entitu, implementuje rozhraní DatabaseEntity obsahující dvě statické metody: `defineMySQLTable` pro samotnou definici databázové tabulky a `getMySQLTableDefinition` pro její získání.

Jmenný prostor `DatabaseUtils` obsahuje celou řadu tříd, které reprezentují strukturu databázových tabulek, položek a dalších vlastností. Jejich výčet je zobrazen na následujícím seznamu:

- `DatabaseTable` – Třída, která reprezentuje databázovou tabulku. Obsahuje název, kolekci se všemi položkami, vlastnosti, unikátní indexy, cizí klíče. Zachycuje M:N vztahy s ostatními tabulkami.
- `Collate` – Obsahuje konstanty definující kódování.
- `ColumnOptions` – Obsahuje konstanty s vlastnostmi, které může mít databázová položka.
- `Constraint` – Slouží pro modelování vztahů 1:1 a 1:M mezi tabulkami. Definuje název tabulky, cizí klíč a akce onUpdate a onDelete (definovány konstantami třídy `ConstraintOptions`).
- `DataTypes` – Obsahuje konstanty s datovými typy, kterých může nabývat datová položka tabulky.
- `DefaultValue` – Obsahuje konstanty s výchozími hodnotami pro datovou položku tabulky.
- `MimeTypes` – Obsahuje konstanty s MIME typy pro datové položky tabulky.
- `StorageEngine` – Obsahuje konstanty pro definici formátu úložiště dat databázové tabulky.

- `TableColumn` – Třída reprezentující datovou položku tabulky. Obsahuje název, datový typ, délku, výchozí hodnotu, kódování, vlastnosti, typ indexu, možnost automatického inkrementování, MIME typ a vlastnost, jestli je položka povinná.
- `TableIndexes` – Definuje typy indexů tabulky.
- `TableOptions` – Definuje vlastnosti tabulky (typ úložiště a kódování).

Definice databázové tabulky může vypadat následovně:

```
//database table name and column names
const TABLE_NAME = "childrens";
const COLUMN_FIRST_NAME = "first_name";
const COLUMN_LAST_NAME = "last_name";

//foreign keyes
const COLUMN_NURSERY_CLASS_ID = "nursery_class_id";

public static function defineMySQLTable() {
  $columns = Array(
     new \Bs\DatabaseUtils\TableColumn(
       $name = self::COLUMN_FIRST_NAME,
       $type = \Bs\DatabaseUtils\DataTypes::VARCHAR,
       $length = 100, $default = null, $collate = null,
       $options = null, $null = false, $index = null
       , $autoIncrement = null, $mimeType = null
     ),
     new \Bs\DatabaseUtils\TableColumn(
       $name = self::COLUMN_LAST_NAME,
       $type = \Bs\DatabaseUtils\DataTypes::VARCHAR,
       $length = 100, $default = null, $collate = null,
       $options = null, $null = false, $index = null,
       $autoIncrement = null, $mimeType = null
     ),
  );

  $options = new \Bs\DatabaseUtils\TableOptions(
     \Bs\DatabaseUtils\StorageEngine::INNO_DB,
     \Bs\DatabaseUtils\Collate::UTF8_CZECH_CI);

  $constraints = Array(
     new \Bs\DatabaseUtils\Constraint(
       $name = "nursery_class",
       $foreignKey = self::COLUMN_NURSERY_CLASS_ID,
       $refenceTable = \Bs\Model\NurseryClass::TABLE_NAME,
       $referenceColumn = "id",
       $onUpdate = \Bs\DatabaseUtils\ConstraintOptions::CASCADE,
       $onDelete = \Bs\DatabaseUtils\ConstraintOptions::RESTRICT,
       $optional = true),
  );

  $manyToManyTableName = Array(
     Array(self::TABLE_NAME, TextNote::TABLE_NAME),
  );

  self::$tableDefinition = new \Bs\DatabaseUtils\DatabaseTable(
     self::TABLE_NAME, $columns, $options,
     $constraints, $manyToManyTableName);
}
```

Informace o názvu databáze a další nastavení se definují v konfiguračním souboru. Tyto informace se v bootstrap souboru načtou a vytvoří se instance třídy `DatabaseGenerator`. Ta nejdříve zkontroluje, jestli na databázovém serveru existuje databáze s definovaným názvem. Pokud ne, tak zahájí generování databáze. Postupně vytvoří všechny tabulky a vygeneruje vazební tabulky pro definované M:N vztahy.

V některých případech se může stát, že v databázi potřebujeme udržovat data, se kterými se v aplikaci nebude pracovat (např. tabulky pro logování). Aby se v aplikaci nemusel zbytečně vytvářet datový model, má programátor možnost umístit do konfiguračního souboru cestu k SQL skriptu, jehož kód se v generátoru databáze načte a provede.

Po ukončení generování struktury dojde k vytvoření instance třídy `BasicProvisioner`, která se stará o naplnění databáze daty. Pokud je aplikace v testovacím režimu (nastaveno v konfiguračním souboru) dojde k vygenerování testovacích dat. V opačném případě se naplní pouze číselníky.

Kromě změny struktury databáze může také dojít k přejmenování datových položek, nebo tabulek (takzvaný refactoring). Aby se tento úkon nemusel provádět ve všech dotazech na databázi, rozhodl jsem se pro použití konstant. Každý datový model tedy definuje konstanty pro názvy datových položek a pro název tabulky. Tyto konstanty se používají ve všech dotazech. V případě potřeby tedy stačí změnit název pouze na jednom místě a následně znovu vygenerovat databázi.

3.3 Bezpečnost aplikace

3.3.1 Cross Site Scripting

Útok, který je nebezpečný hlavně pro veřejně přístupné webové stránky. Jak na svém blogu píše Jakub Vrána [7], útočník se snaží vložit do systému textový řetězec, který obsahuje spustitelný kód (například v jazyce JavaScript), nebo HTML značky. Takto vložený řetězec se poté vypíše uživateli a dojde ke spuštění kódu. Může přitom dojít k zaslání citlivých dat na server útočníka, nebo vypsání falešného přihlašovacího formuláře.

Toto riziko ve frameworku řeší třída `Form`. Každý vstupní řetězec odeslán uživatelem z formuláře je automaticky ošetřen funkcí `htmlSpecialChars`, která všechny speciální znaky (např. znak <) převede na html entity. Uživateli se poté zobrazí korektně, ale nemohou způsobit bezpečnostní riziko.

3.3.2 SQL Injection

Útok typu SQL Injection [8] může nastat v případě, kdy se do SQL dotazu předává parametr, který do systému zadává uživatel – například pomocí formulářového pole, nebo parametru v url adrese. Útočník tak může přímo modifikovat SQL příkaz a může například dojít k vypsání všech záznamů z databáze.

Aby se zabránilo tomuto zneužití, je nutné každý předávaný argument do dotazu ošetřit funkcí `real_escape_string`, která nebezpečné znaky ošetří zpětným lomítkem. Aby nemusel programátor tuto funkci volat ručně v každém dotazu a snížilo se zároveň riziko, že na toto ošetření zapomene, je tato úloha řešena automaticky ve třídě `Mysql`. Všechny předané parametry jsou při sestavování výsledného SQL dotazu ošetřeny.

3.3.3 Cross-Site Request Forgery

Tento typ útoku se nejlépe vysvětlí na příkladu, který uvádí Jakub Vrána [9]. Představme si, že budeme mít skript na smazání záznamu z databáze, kterému se budou předávat parametry pomocí url adresy (např. `delete.php?id=1`). Tento skript poté provede ověření, jestli je uživatel přihlášen, pokud ano, tak smaže záznam s daným identifikátorem. Pokud bude útočník znát název skriptu včetně parametrů, může tuto adresu podstrčit na své stránky – například jako obrázek, který má jako zdroj uvedenou adresu na smazání záznamu. Pokud bude uživatel přihlášen a pro informace o přihlášení se používá cookie (v PHP výchozí použití session proměnných), tak dojde k provedení skriptu a smazání záznamu, aniž by o tom uživatel věděl.

Vzhledem k návrhu aplikace by k této situaci mohlo dojít pouze u mazání záznamů. Tato bezpečnostní slabina je vyřešena pomocí předávání bezpečnostního řetězce. Ten se náhodně vygeneruje po přihlášení uživatele do systému a uloží se do session proměnné. Každý odkaz na smazání záznamu tento řetězec obsahuje jako další parametr. Ve skriptu pro smazání záznamu dochází ke kontrole, jestli je řetězec z url adresy shodný s řetězcem uloženým v session. Vzhledem k tomu, že je řetězec unikátní pro každé přihlášení, nemá jej útočník šanci zjistit.

3.3.4 Session fixation

Jak uvádí Jakub Vrána [10], útok spočívá v tom, že útočník podvrhne oběti vlastní session token a následně počká, až se oběť přihlásí do systému. Vzhledem ke znalosti tokenu se poté může začít za oběť vydávat a získá tím plný přístup k systému. Ošetření tohoto rizika je poměrně jednoduché, spočívá v zavolání PHP funkce `session_regenerate_id()`, která uživateli přidělí nový identifikátor. Ve frameworku dochází k volání této funkce ve třídě `Authentication` po přihlášení uživatele.

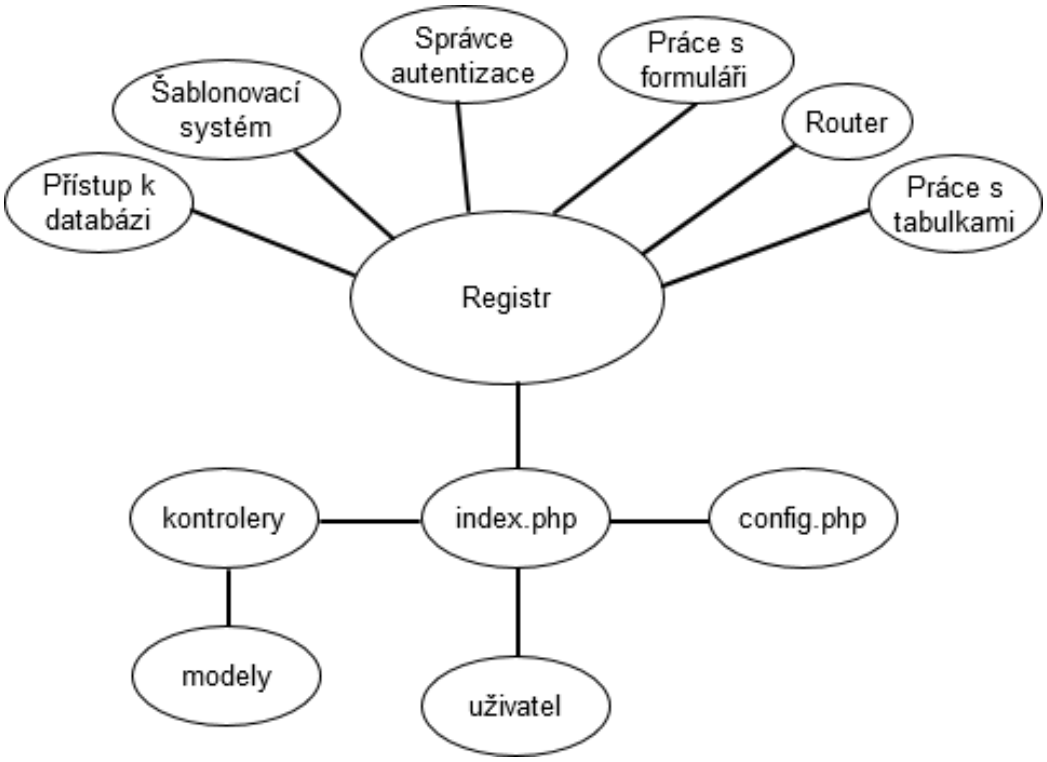
3.3.5 Man in the middle

Útok typu man in the middle [11] (v překladu člověk uprostřed) je jeden z nejznámějších problémů v kryptografii. Komunikace mezi klientem a serverem běžně probíhá na protokolu http, který přenáší data v textové podobě. Pokud by se útočník vložil mezi klienta a server, mohl by veškerá data odposlouchávat a měnit. Řešení spočívá v použití šifrovaného protokolu https a SSL certifikátu. Veškerá přenášená data jsou poté šifrovaná a certifikát zároveň zajistí autenticitu obou protistran.

3.4 Struktura aplikace

Jak již bylo zmíněno v kapitole 3.2.4, aplikace využívá návrhový vzor FCP. Uživatel tedy vždy přistupuje k souboru index.php. V tom dochází k vytvoření instance registru a následnému načtení nastavení ze souboru `config.php`. V registru se vytvoří instance všech dostupných částí frameworku. Pokud na databázovém serveru neexistuje vytvořená databáze, tak dojde ke spuštění generátoru databáze, jejímu naplnění daty a instalaci všech kontrolerů (viz kapitola 3.4.4). Po dokončení těchto úkonů dojde k předání řízení směrovači (třída `Router`).

Pokud je třída závislá na jiné třídě, musí před jejím použitím dojít k vložení souboru, ve kterém je externí třída definovaná. Aby nebylo nutné tento úkon provádět ručně ve všech částech aplikace, definuji v bootstrap souboru metodu `__autoload`, která se volá při použití neznámé třídy. Metodě je předán název třídy, podle kterého se pokusí nalézt a importovat zdrojový soubor. V celé aplikaci používám tyto jmenné konvence: zdrojový soubor se třídou má formát `NazevTridy.class.php` a rozhraní má formát `NazevRozhrani.interface.php`. Z názvu třídy se tedy sestaví odpovídající název souboru. Ten se importuje s využitím direktivy `include_path` obsahující všechny cesty ke zdrojovým kódům aplikace.



Obrázek 1 – struktura aplikace

3.4.1 Adresářová struktura aplikace

- controllers
- database_utils – sada tříd starajících se o generování databáze
- locale – databázové soubory s překlady pro různé jazykové mutace

- model
  - childrens
  - persons
  - trips
  - utilities
- registry
  - objects
    - database
    - forms
- services
  - i18n
- tests
- users_data
- views
  - default – výchozí vzhled aplikace
    - css
    - img
    - templates
  - defaultForms – výchozí šablony pro formuláře
  - filterForms – šablony pro formuláře, které využívá filtrační formulář
  - js – JavaScriptové soubory
  - plugins – JavaScriptové doplňky třetích stran (např.: jQuery)

3.4.2 Modely

Modely aplikace neobsahují žádnou aplikační logiku, pouze definují strukturu entity a její databázovou reprezentaci. Všechny třídy se nacházejí ve jmenném prostoru `Model`, který je ještě členěn na jmenné prostory `Model\Childrens`, `Model\Persons` a `Model\Trips` – toto členění se odráží i v adresářové struktuře. Všechny modely dědí od abstraktní třídy `BasicEntity`, která implementuje rozhraní `DatabaseEntity`.

3.4.3 Servisní třídy

Ke každému modelu aplikace přísluší servisní třída, která se stará o mapování databázových dat na daný objekt a zajišťuje všechny operace s objektem. Všechny servisní třídy dědí od abstraktní třídy `AbstractDataService`, která implementuje rozhraní `DataService`. Rozhraní `DataService` definuje metody pro načítání, ukládání, úpravu a mazání záznamů. Kromě těchto základních operací se v servisních třídách definují všechny metody pro práci jak se samotným objektem, tak s jeho kolekcí.

3.4.4 Kontrolery

Kontroler obecně přijímá požadavky od uživatele, načítá/upravuje data z modelu a sestavuje pohledy. Aplikace používá jednoduchou adresní strukturu, kde je za lomítkem umístěn název modulu, za dalším lomítkem akce daného modulu, za kterou následují předávané parametry oddělené lomítky (např.: `/ChildrensController/edit/4/`). Každý modul aplikace má tedy vlastní kontroler. Všechny kontrolery dědí od abstraktní třídy `AbstractController`.

Po dokončení směrování, předá směrovač řízení instanci třídy `AdminController`, ve které nejdříve dochází k ověření autenticity uživatele – pokud uživatel není přihlášen, sestaví kontroler přihlašovací formulář. Stará se také o sestavení grafického rozhraní programu: generování menu a vložení základních proměnných do šablony. Kontroler následně z url adresy dekóduje název modulu a předá řízení příslušnému kontroleru.

Ve třídy `AdminController` se odchytává výjimka `PageDoesntExist`, která je vyvolána v případě, že je v adrese předán neplatný název modulu. Každá akce kontroleru je zároveň uživatelským oprávněním. Tyto oprávnění se kontrolují v kontroleru, a pokud uživatel nemá dostatečné oprávnění, je vyvolána výjimka `AuthorizationException`. Ta je odchycena ve třídě `AdminController` a uživateli je vypsaná chybová hláška. Oprávnění přihlášeného uživatele jsou uchovávány v instanci třídy `User`, na kterou se lze odkázat z instance `Authentication` uložené v registru.

Aby bylo možné udržovat v databázi informace o uživatelských oprávněních, musíme mít v databázi uložené akce všech kontrolerů. Z tohoto důvodu definuje každý kontroler konstanty obsahující jméno modulu a názvy všech akcí. V souboru `config.php` je definována kolekce se všemi moduly aplikace a jejich uživatelsky přívětivými popisky. Z této kolekce je následně sestavováno navigační menu a pomocí třídy `ServicesInstaller` dojde k naplnění všech dostupných oprávnění do databáze.

V konstruktoru každého kontroleru se z adresy dekóduje akce a zavolá se příslušná metoda. V metodách jednotlivých akcí dochází k sestavení výstupní šablony, generování formulářů a volání servisních tříd pro načítání dat a manipulací s nimi.

3.4.5 Pohledy

Pohledy jsou sestavovány šablonovacím systémem ze souborů tpl. Ty jsou umístěny v adresáři `templates` – šablony pro výchozí vzhled aplikace jsou umístěny ve složce default. Systém je navržen tak, že pokud by do budoucna bylo potřeba vytvořit jiný layout aplikace (například optimalizovaný pro mobilní zařízení), tak se pouze vytvoří nová sada šablon a příslušné kaskádové styly.

Ve složce `templates` jsou také umístěny šablony pro formulářové prvky. Všechny stránky aplikace jsou nastylovány pomocí sady kaskádových stylů Twitter Bootstrap, které usnadňují sestavování grafického rozhraní. Vzhled stránek je zároveň optimalizovány

pro všechny prohlížeče a rozlišení obrazovky. V prezentační části aplikace je také používán JavaScriptový framework jQuery, která se stará o usnadnění a zpříjemnění uživatelského prostředí. V některých formulářích je používáno asynchronní komunikace pomocí Ajaxu.

3.5 Internacionalizace a lokalizace aplikace

Internacionalizace je způsob vytváření zdrojového kódu tak, aby bylo do budoucna možné provozovat aplikaci v libovolné jazykové mutaci bez další změny programu. Lokalizace poté obnáší vytváření databáze překladů pro danou jazykovou verzi.

I když je aplikace primárně určená pro provoz v českém jazyce, je rozhodně dobré myslet do budoucna a být připraven na situaci, kdy bude potřeba přejít na jinou jazykovou variantu. Pro internacionalizaci existuje několik nástrojů, já jsem zvolil z mého pohledu asi nejlepší variantu a to GNU gettext. Jedná se o sadu nástrojů, které byly vyvinuty v projektu GNU a původně byly určeny pro internacionalizaci unixového systému. V současnosti existuje implementace pro všechny běžné programovací jazyky včetně PHP:

Gettext pracuje na následujícím principu: všechny textové řetězce, které se mohou vypsat uživateli, jsou psány v angličtině a jsou obaleny speciální funkcí. Po takovéto přípravě zdrojových kódu dojde k vygenerování všech řetězců do překladového souboru. Anglicky napsaný řetězec poté tvoří unikátní klíč, který je možné přeložit do libovolného jazyka. Funkce, kterou jsou textové řetězce obaleny, nejdříve zjistí aktuální jazykovou verzi programu, pokud existuje překladový soubor pro tento jazyk s daným textem, vrátí přeloženou verzi textu. V opačném případě vrátí originální text. Nástroj nabízí dvě verze těchto funkcí a to pro překlad jednotného a množného čísla. Funkci pro překlad množného čísla se poté předá ještě počet, podle kterého se vybere správně skloňovaný řetězec.

3.5.1 Lokalizace v kódu

V jazyce PHP jsou výše zmiňované funkce přímo implementovány: pro jednotné číslo existuje funkce `gettext` (nebo její alias ve formě podtržítka) a pro množné číslo jazyk nabízí funkci `ngettext`. Problém nastává v případech, kdy potřebujeme do textového řetězce předat nějaký parametr. Máme na výběr dvě možnosti: buď řetězec rozdělit, vložit mezi něj parametr a překládat každou část zvlášť, nebo místo parametru zapsat zástupný znak. První způsob je velmi nešikovný, pro překladatele by působil velmi zmateně a s velikou pravděpodobností by došlo k chybnému překladu.

Z tohoto důvodu jsem vytvořil třídu `Text` umístěnou ve jmenném prostoru `Services\i18n`, která nabízí dvě statické metody. Metoda `tr` slouží pro překlad jednotného čísla, předává se jí samotný řetězec a pole obsahující parametry. Druhá metoda se jmenuje `trn`, ta přijímá textový řetězec v jednotném a množném čísle, počet (podle kterého se vybere verze se správným skloňováním) a pole s parametry. Všechny parametry se v řetězcích nahradí zástupným znakem `%s`. Překlad řetězců může vypadat následovně:

```
Bs\Services\i18n\Text::tr("Hello %s %s.",
   Array($firstName, $lastName));

Bs\Services\i18n\Text::trn("%s dog in %s", "%s dogs in %s",
   $count, Array($count, $location));
```

3.5.2 Lokalizace v šablonách

Většina textových řetězců, které se zobrazují uživateli, se nachází v šablonových souborech. Z tohoto důvodu bylo nezbytné doplnit šablonovací systém o podporu internacionalizace. Vytvořil jsem proto dva speciální tagy, které se při parsování výstupu naleznou a jejich obsah je předán metodám třídy `Text`. Překlad v šablonovacím souboru vypadá následovně:

```
<!--tr t=" Hello %s %s." p="{firstName}&{lastName}"-->

<!--trn t="%s dog in %s" pt="%s dogs in %s"
   n="{count}" p="{count}&{location}"-->
```

3.5.3 Práce s datem

Další částí aplikace, která je závislá na konkrétní jazykové mutaci, jsou data. Pro práci s nimi slouží další součást jmenného prostoru `Services\i18n` – třída `DateUtil`. Ta slouží pro převod dat mezi jednotlivými formáty. Všechny datové údaje jsou před uložením do databáze převedeny do časové zóny UTC a po opětovném načtení převedena do aktuálního časového pásma.

4 Návrh databáze

V aplikaci využívám databázový systém MySQL, pro který jsem implementoval třídu Mysql dědící od třídy Database a generátor databáze. Celá struktura databáze je definována v jednotlivých modelech s tím, že relační tabulky pro vztahy M:N se generují automaticky.

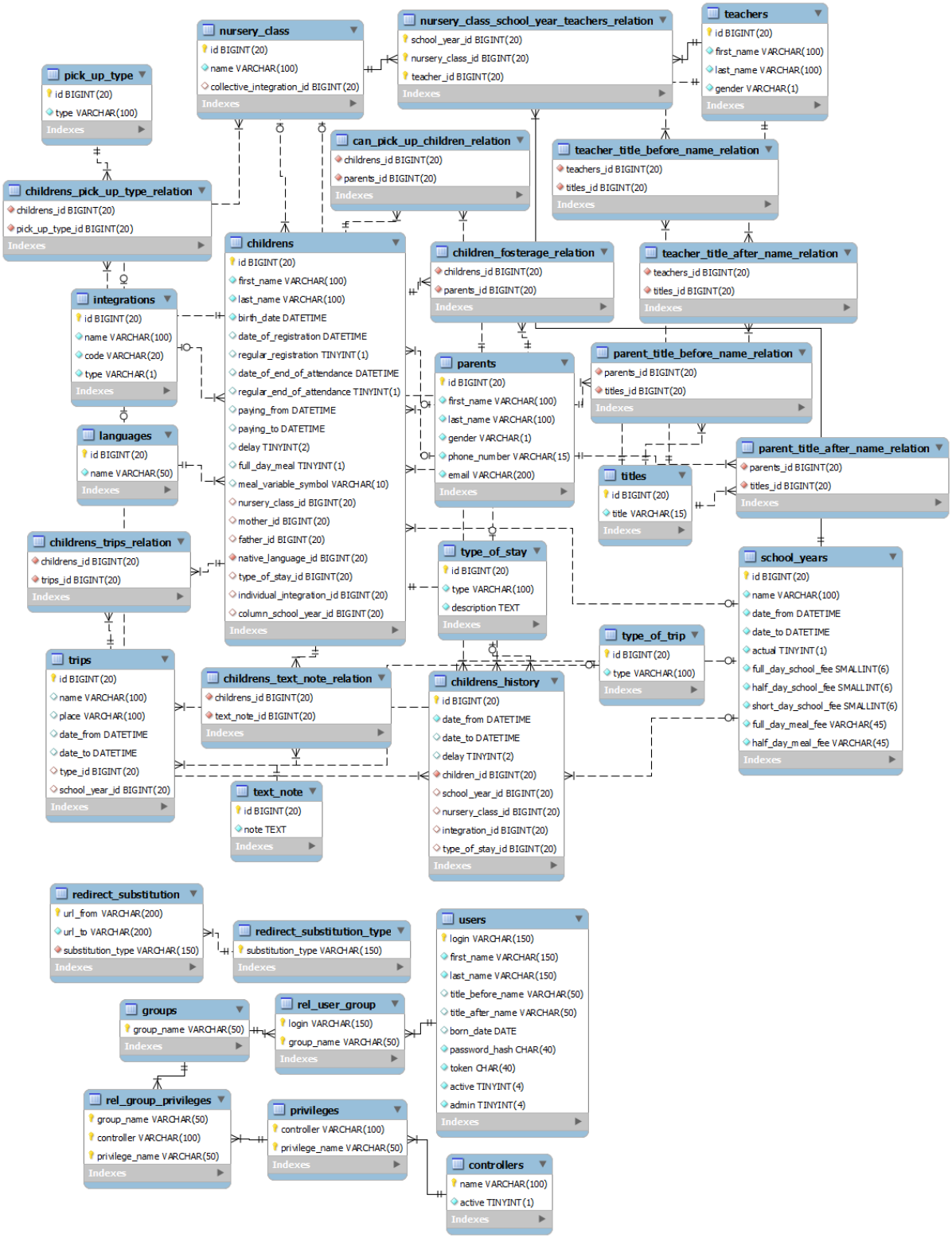
Aby se na straně aplikace a frameworku usnadnila práce s databázovými tabulkami, používá každá z tabulek umělý primární klíč nazvaný ID. Z pohledu návrhu databáze není tento způsob správný – mělo by se využívat přirozených primárních klíčů a volba umělého primárního klíče by se měla použít pouze v případě, že žádný jednoznačný primární klíč neexistuje. Tím by se však velmi zkomplikovala aplikační vrstva. Proto jsem dospěl k tomuto kompromisu.

4.1 Struktura databáze

Výsledná databáze obsahuje 32 tabulek. Vzhledem k tomu, že je ER diagram této databáze již poměrně rozsáhlý (Obrázek 2), ve stručnosti zde popíši nejdůležitější tabulky:

- can_pick_up_children_relation – Relační tabulka znázorňující vztah mezi dítětem a osobou, která jej může vyzvedávat.
- groups – Skupiny uživatelů, kterým lze přiřazovat jednotlivá oprávnění.
- children_fosterage_relation – Zachycuje svěření dítěte do péče.
- childrens – Zachycuje všechny potřebné informace o dětech mateřské školy.
- childrens_history – Relační tabulka zaznamenávající historii o pobytu dítěte.
- childrens_pick_up_type_relation – Typ osoby, která může dítě vyzvednout.
- childrens_text_note_relation – Přiřazuje textové poznámky k dítěti.
- childrens_trips_relation – Vazební tabulka realizující vztah mezi dítětem a školním pobytem.
- integrations – Číselník všech typů integrací.
- languages – Číselník jazyků.
- nursery_class – Uchovává všechny třídy školky.
- nursery_class_school_year_teachers_relation – Přiřazuje školní rok ke třídě.
- parent_title_after_name_relation – Přiřazuje tituly za jménem rodiči dítěte.
- parent_title_before_name_relation - Přiřazuje tituly před jménem rodiči dítěte.
- parents – Tabulka s rodiči dětí.
- pick_up_type – Číselník typů osob ve vztahu k dítěti.
- privileges – Uživatelská oprávnění k jednotlivým modulům aplikace.
- school_years – Zachycuje všechny školní roky, které jsou v systému uchovány.
- teacher_title_after_name_relation – Přiřazuje tituly za jménem učiteli.
- teacher_title_before_name_relation – Přiřazuje tituly před jménem učiteli.
- teachers – Slouží pro uchování všech učitelů školky.
- text_note – Textová poznámka, kterou lze přiřadit k dítěti.

- titles – Číselník akademických titulů.
- trips – Vytvořené školní pobyty s přiřazeným druhem.
- type_of_stay – Číselník typů pobytu dítěte ve školce.
- type_of_trip – Číselník druhů školních pobytů.
- users – Obsahuje informace o všech uživatelích systému.



Obrázek 2 – struktura databáze

5 Funkcionalita aplikace

Aplikace je rozdělena na několik funkčních modulů, které jsou zobrazeny v horizontálním navigačním menu. Po vstupu do modulu se v levém panelu otevře vertikální menu, ve kterém jsou zobrazeny základní akce modulu. Mezi ně patří výpis všech záznamů příslušejících k danému modulu a vložení nového záznamu. Pokročilé moduly nabízejí ještě další akce, které budou podrobně probrány dále.

Po výběru vypsání existujících záznamů dojde k zobrazení tabulky (Obrázek 3) obsahující základní údaje o zobrazovaných entitách s přímým odkazem na editaci a smazání záznamu. Tuto tabulku lze řadit podle libovolných atributů vzestupně a sestupně – to je zajištěno pomocí JavaScriptového pluginu TableSorter.

Příjmení	Jméno	Telefon	Email	Děti	
Gryč	Milan	462212456	Milan.Gryc@email.cz	Larisa Gryčová	Upravit Smazat
Gryčová	Jana	618815104	Jana.Grycova@email.cz	Larisa Gryčová	Upravit Smazat
Havel	Daniel	570393880	Daniel.Havel@email.cz	Božena Havlová	Upravit Smazat
Havlová	Božena	132812499	Bozena.Havlova@email.cz	Božena Havlová	Upravit Smazat

Obrázek 3 – ukázka výpisu seznamu rodičů

Po kliknutí na možnost vložení nového záznamu je uživatel přesměrován na stránku s formulářem obsahující všechny informace o daném objektu. Na každém formuláři je také možnost zůstat na kartě nově vytvořeného záznamu. Pokud uživatel tuto možnost nezaškrtne, tak dojde k přesměrování na výpis všech záznamů. Po odeslání formuláře dojde k jeho validaci. Pokud není správně vyplněn, je o tom uživatel informován, do formuláře se při tom automaticky vyplní odeslané hodnoty.

Po kliku na tlačítko pro smazání záznamu je uživatel nejdříve upozorněn dialogovým oknem a ke smazání záznamu dojde až po jeho potvrzení.

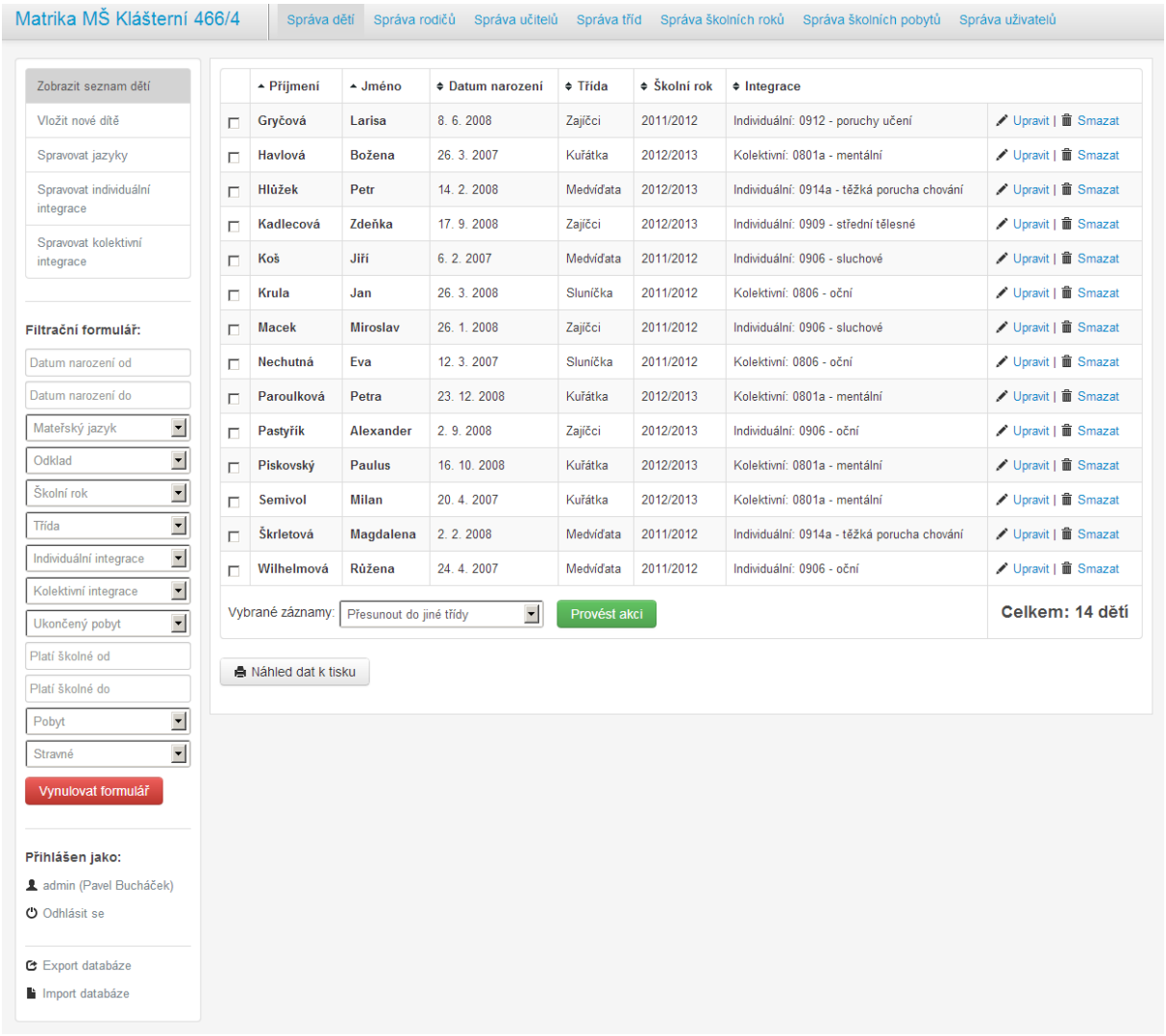
O úspěchu a případných chybách je uživatel vždy informován pomocí zpráv zobrazovaných pod hlavním navigačním menu.

5.1 Správa dětí

5.1.1 Zobrazení seznamu dětí

Správa dětí tvoří nejdůležitější a zároveň nejobsáhlejší část aplikace. Seznam s výpisem všech uchovávaných dětí (Obrázek 4) je doplněn o filtrační formulář, který je umístěn v levém panelu pod navigačním menu. Po výběru libovolné položky dojde k automatickému odeslání a uživateli jsou zobrazeny pouze záznamy, které odpovídají nastavenému filtru. Celkový počet zobrazených záznamů je zobrazen pod posledním řádkem tabulky.

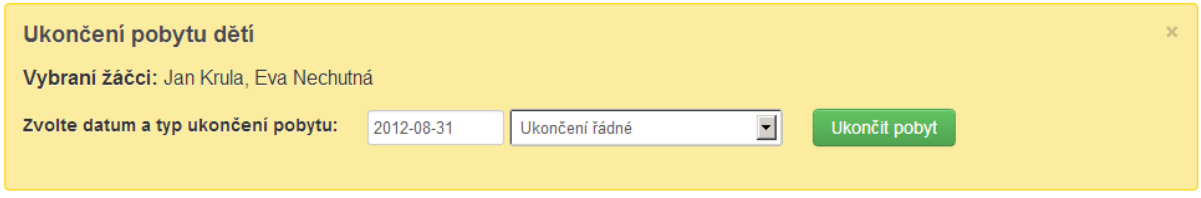
Takto vyfiltrovaná data je rovnou možné tisknout. Před samotným tiskem je uživateli zobrazen náhled dat k tisku, kde má možnost záznamy seřadit podle libovolných atributů. Na tiskové sestavě se v záhlaví zobrazují informace o mateřské školce a pod seznamem dětí je zobrazena informace o celkovém počtu záznamů a použitém filtračním formuláři.



Obrázek 4 – seznam dětí

5.1.2 Hromadné akce

Ve správě dětí je také možné vybírat libovolný počet zobrazených záznamů, na které lze následně aplikovat takzvané hromadné akce (Obrázek 5). Těmi jsou: přesunutí dětí do jiné třídy, přesunutí do nového školního roku, ukončení pobytu a smazání záznamů.



Obrázek 5 – ukázka hromadného ukončení pobytu

5.1.3 Karta dítěte

Karta dítěte (Obrázek 6) je členěna na čtyři sekce, ke každé sekci je přitom možné přiřazovat uživatelská oprávnění. Každá sekce je reprezentování jednou záložkou, mezi záložkami může uživatel přepínat.

Záložka základní údaje obsahuje následující údaje:

- příjmení,
- křestní jméno,
- datum narození,
- mateřský jazyk,
- typ zápisu (řádný/během roku),
- odklad (žádný, první, druhý),
- datum ukončení pobytu,
- aktuální školní rok,
- třída, do které je dítě aktuálně zařazeno,
- typ integrace,
- matka a otec,
- svěření do péče,
- informace o tom, kdo dítě vyzvedává.

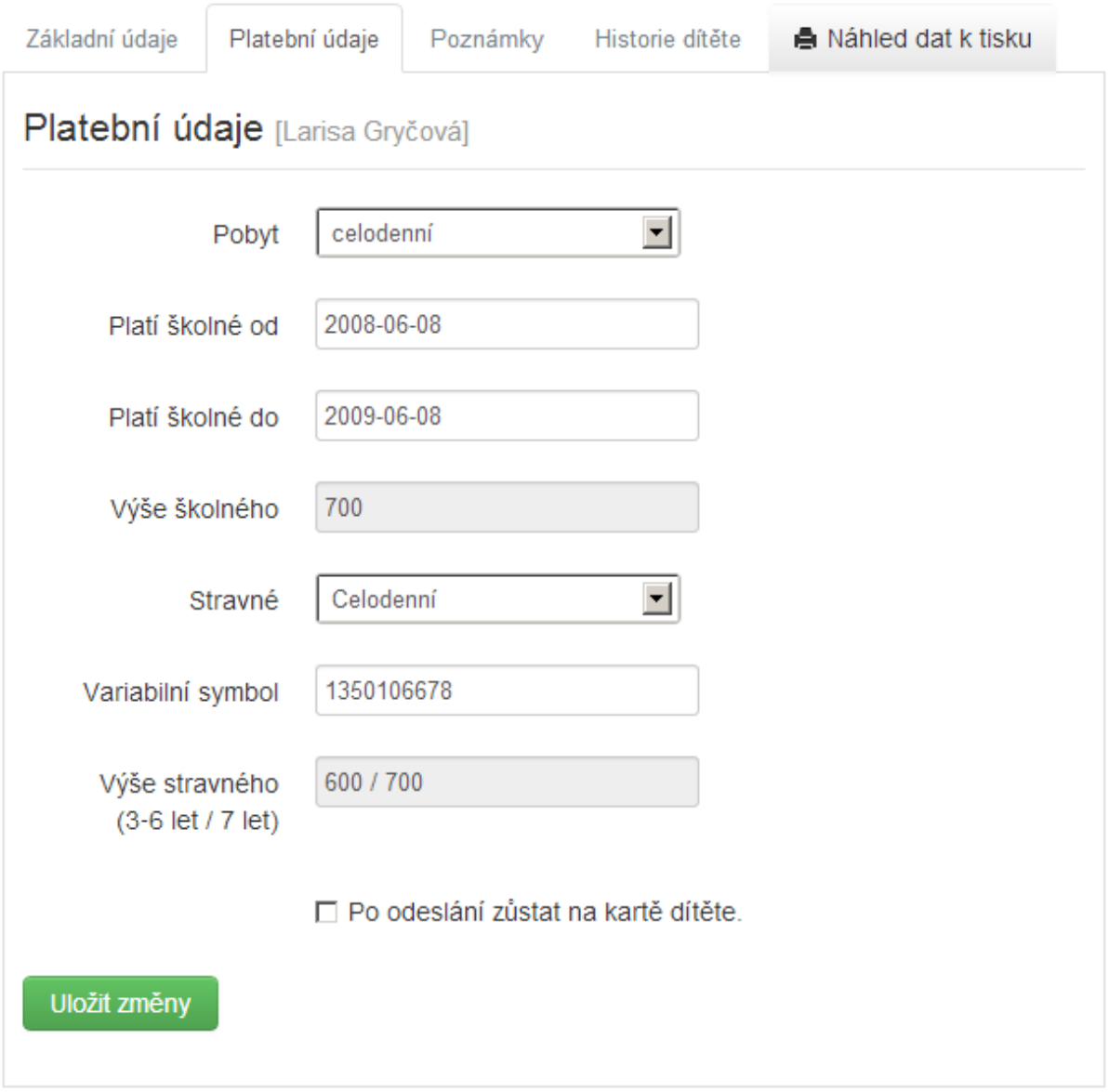
Záložka platební údaje obsahuje:

- typ pobytu (celodenní, polodenní, zkrácený),
- data od kdy, do kdy dítě platí školné,
- výše školného pro aktuální školní rok,
- typ stravného (celodenní, polodenní),
- variabilní symbol pro placení stravného,
- výše stravného (pro děti ve věku 3-6 let a pro 7leté děti)

V záložce poznámky je možné ke každému dítěti vkládat libovolný počet textových poznámek, upravovat je a mazat. Pod textovými poznámkami je zobrazen seznam školních pobytů, kterých se dítě zúčastnilo.

Poslední záložkou je historie dítěte. Historie se týká položek školní rok, třída, integrace, typ pobytu a odklad. Při změně libovolného z těchto atributů dojde automaticky k vložení záznamu do tabulky `childrens_history`. Takto uložené záznamy se poté zobrazují společně s daty od-do na kartě každého dítěte.

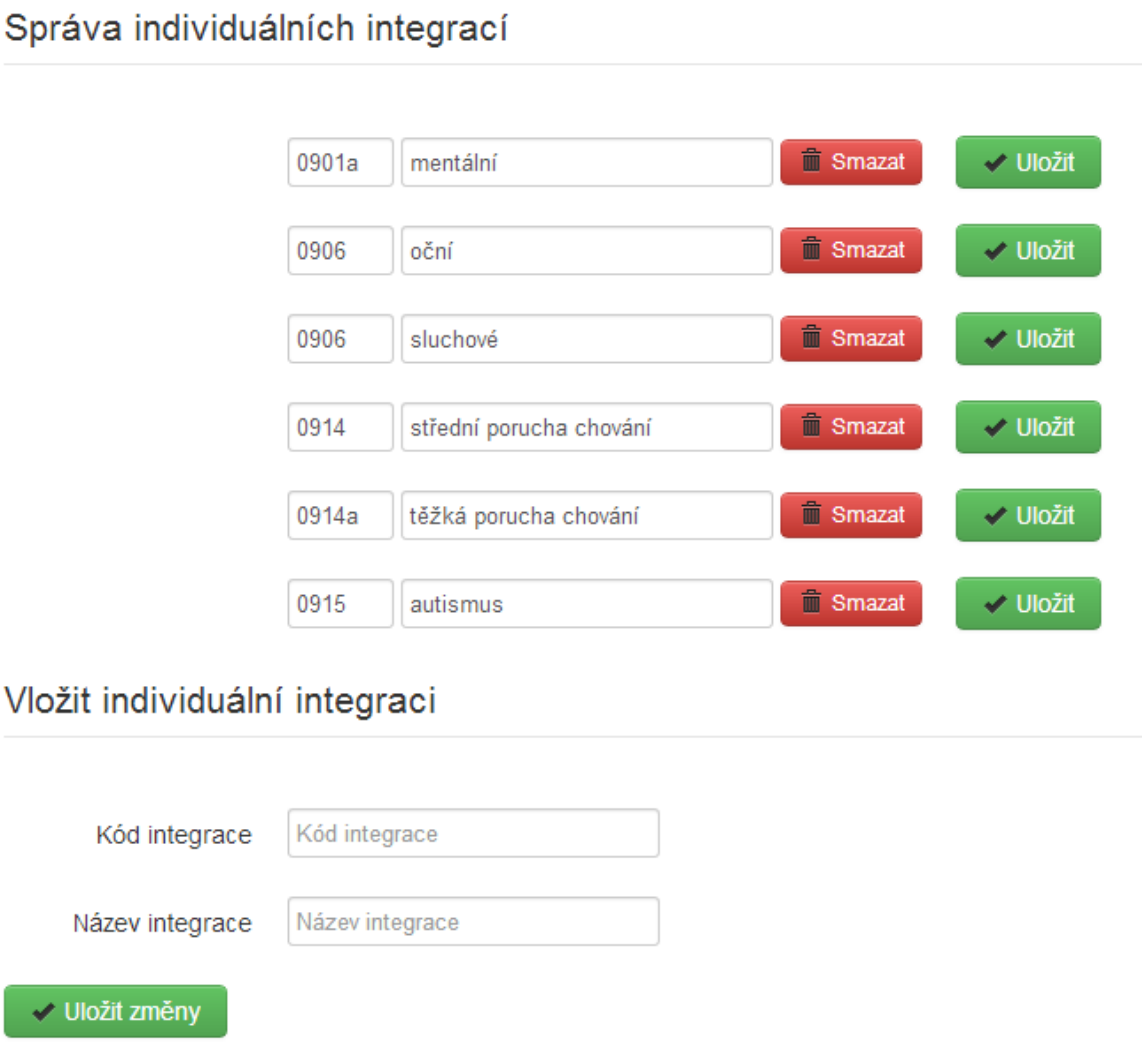
Uživatel má také možnost vytisknout kartu dítěte, na které jsou kromě záhlaví umístěny veškeré výše zmíněné informace.



Obrázek 6 – karta dítěte

5.1.4 Správa integrací a jazyků

Při správě záznamů s dětmi se využívají číselníky jazyků a integrací. U těch se ale může stát, že se v průběhu používání aplikace změní. Proto je nutné dát uživateli možnost je programově upravovat. K tomu slouží modul pro úpravu dynamických číselníků. Obsahuje vždy seznam existujících záznamů s přímou možností jejich editace a smazání. Dále také obsahuje možnost přidání nového záznamu (Obrázek 7).



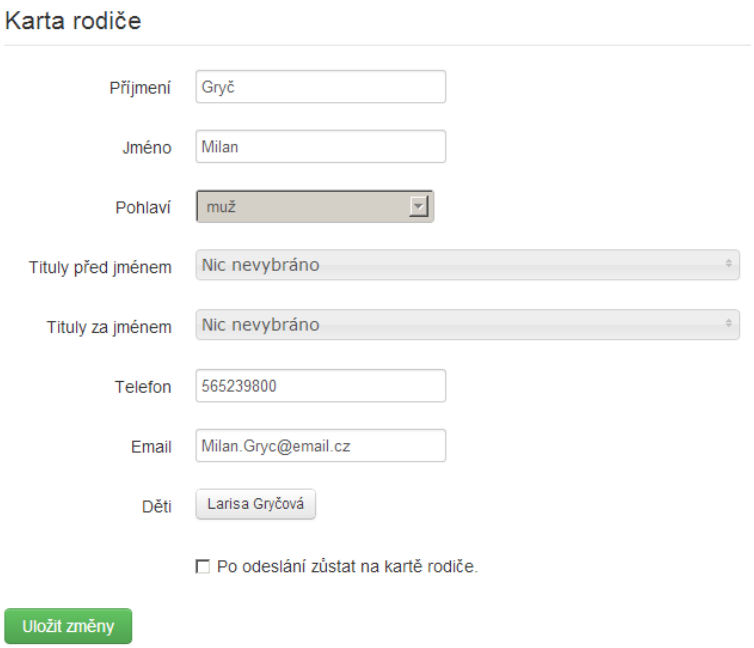
Obrázek 7 – ukázka správy dynamického číselníku itnegrací

5.2 Správa rodičů

O rodičích dětí nám v současné době stačí uchovávat pouze základní kontaktní informace. Formulář pro vytváření a úpravu rodičů (Obrázek 8) tedy obsahuje následující položky:

- příjmení,
- křestní jméno,
- pohlaví,
- tituly před a za jménem (je jich možné vybrat více),
- telefon,
- email.

Na kartě rodiče se dále zobrazují odkazy na karty dětí, které mají rodiče přiřazeny.



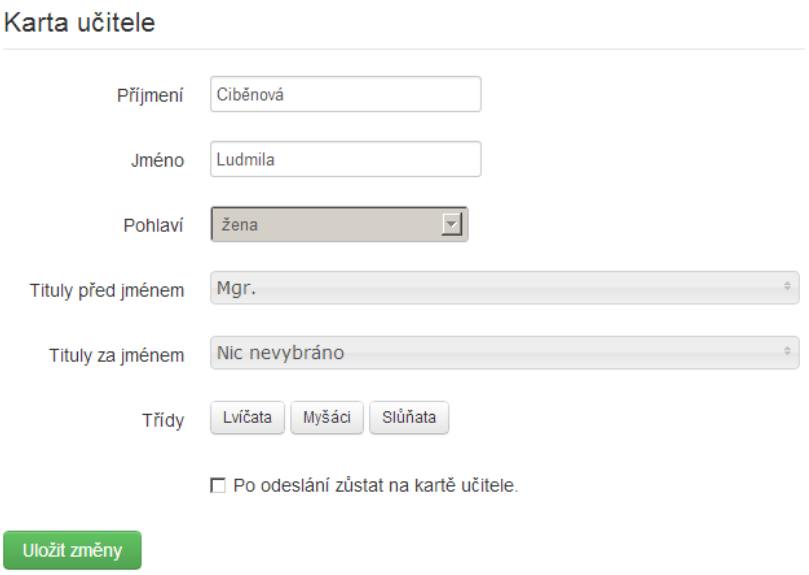
Obrázek 8 – karta rodiče

5.3 Správa učitelů

Ve správě učitelů (Obrázek 9) je situace podobná jako u rodičů. Uchováváme informace:

- příjmení,
- jméno,
- pohlaví,
- tituly před a za jménem.

Pod těmito hodnotami jsou zobrazeny odkazy na karty tříd, ve kterých jsou učitelé v aktuálním školním roce přiřazeni.



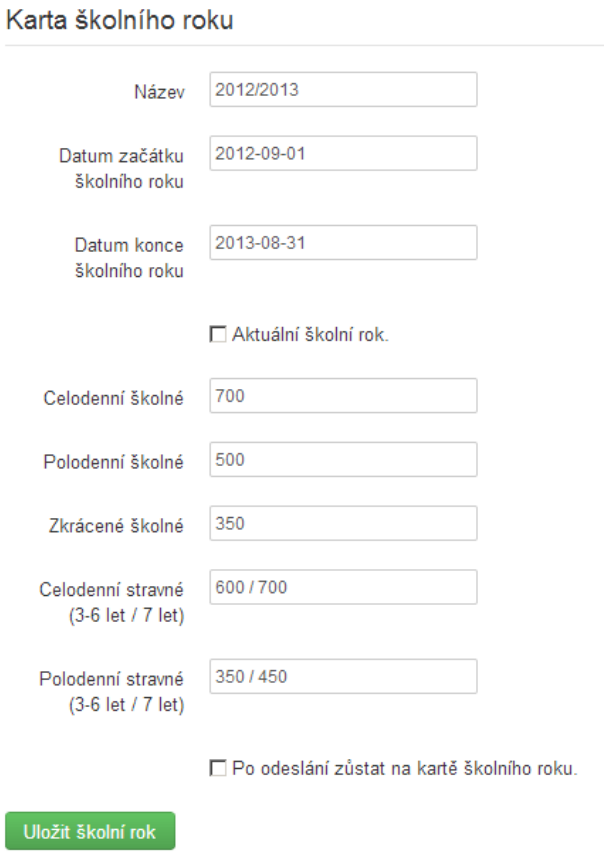
Obrázek 9 – karta učitele

5.4 Správa školních roků

U školního roku (Obrázek 10) potřebujeme udržovat následující údaje:

- název,
- datum začátku a konce školního roku (z důvodu počítání dosaženého věku dítěte v daném školním roce),
- hodnotu celodenního, polodenního a zkráceného školného,
- hodnotu celodenního a polodenního stravného.

Kromě těchto informací můžeme u školního roku nastavit příznak, že se jedná o aktuální školní rok. Tento údaj slouží pro filtrování dat – například v seznamu učitelů se zobrazují třídy, ve kterých učitelé učí – ty jsou brány právě z aktuálního školního roku.



Obrázek 10 – karta školního roku

5.5 Správa tříd

5.5.1 Typ integrace

Kromě jména třídy musíme u třídy uchovávat také typ integrace. Zde mohou nastat dva případy: buď se jedná o třídu s kolektivní integrací (poté mají tuto integraci všechny děti zařazené ve třídě), nebo se jedná o třídu bez integrace (děti v této třídě mohou mít kolektivní integraci).

Pro jednotlivé školní roky, které jsou v systému vedeny, se třídě přiřazují učitelé. U každé třídy může být pro daný školní rok přiřazeno více učitelů (pomocí komponenty Multiselect), učitel může být zároveň přiřazen u více tříd.

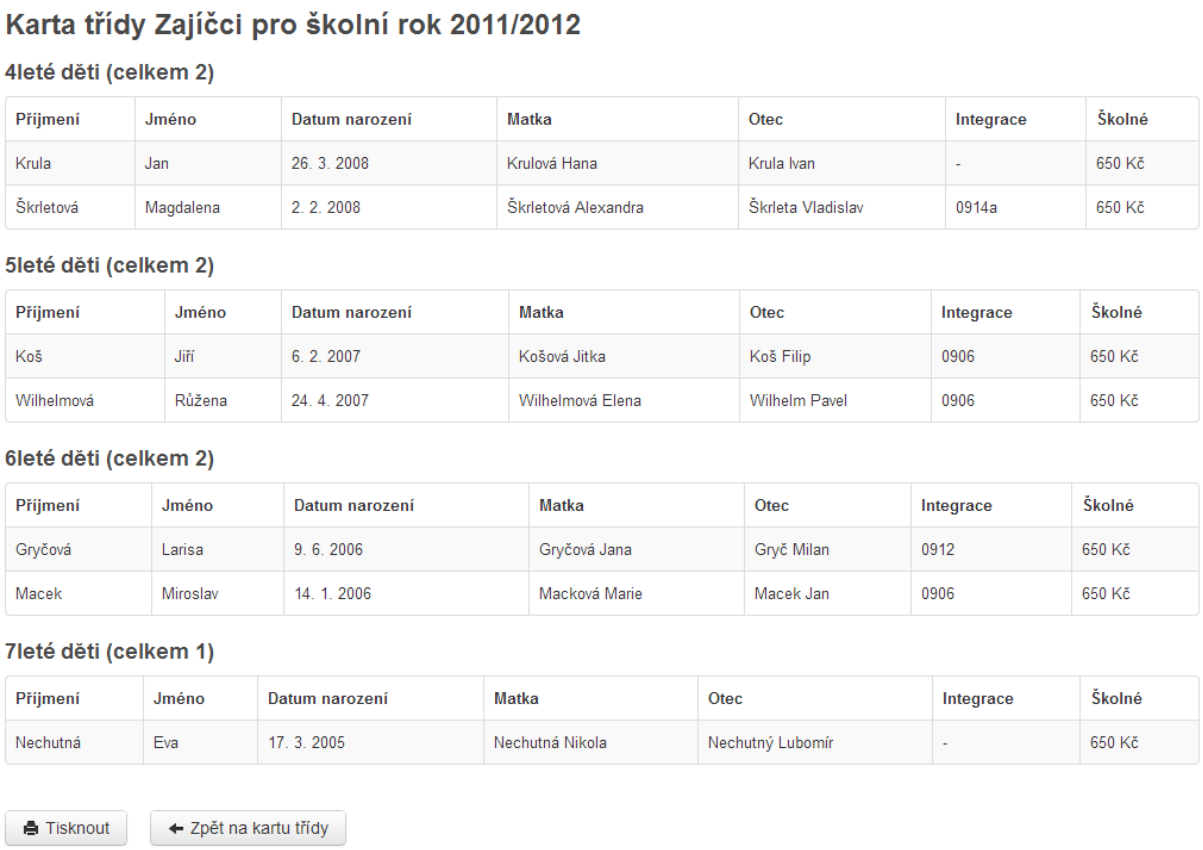
5.5.2 Tiskové sestavy pro jednotlivé školní roky

U každého školního roku je možnost vytvořit tiskovou sestavu se seznamem dětí, které v daném školním roce třídu navštěvovaly (Obrázek 11). Důležité je rozdělení tohoto seznamu dětí podle jejich věku dosaženého v daném školním roce na děti čtyřleté až sedmileté.

O správné načtení takto rozdělených dětí se stará metoda `getChildrensPerAgeForNurseryClass` v servisní třídě `ChildrenService`. Dotaz pro načtení příslušných dat z databáze vypadá následovně:

```
$subselect = $this->db->buildSelect(
   Array(\Bs\Model\Childrens\ChildrenHistory::COLUMN_CHILDREN_ID),
   \Bs\Model\Childrens\ChildrenHistory::TABLE_NAME,
   new \Bs\Registry\Objects\Database\WhereCondition(
      \Bs\Model\Childrens\ChildrenHistory::COLUMN_NURSERY_CLASS_ID
      . " = ? AND " .
      \Bs\Model\Childrens\ChildrenHistory::COLUMN_SCHOOL_YEAR_ID
      . " = ?",
      Array($nurseryClassId, $schoolYearId)
   ),
   \Bs\Model\Childrens\ChildrenHistory::COLUMN_CHILDREN_ID
);

$query = $this->db->buildSelect(
   Array(
      \Bs\Model\SchoolYear::COLUMN_NAME,
      $this->db->pasteGroupConcat(
         \Bs\Model\Childrens\Children::TABLE_NAME . ".id",
         \Bs\Model\Childrens\Children::COLUMN_LAST_NAME, true, ";")
      . $this->db->pasteAs("ids"),
      $this->db->pasteCount(
         \Bs\Model\Childrens\Children::TABLE_NAME . ".id")
      . $this->db->pasteAs("count"),
      $this->db->pasteFloor($this->db->pasteDateDiff(
         \Bs\Model\SchoolYear::COLUMN_DATE_TO,
         \Bs\Model\Childrens\Children::COLUMN_BIRTH_DATE) . " / 365.25")
      . $this->db->pasteAs("age"),
   ),
   \Bs\Model\Childrens\Children::TABLE_NAME,
   new \Bs\Registry\Objects\Database\WhereCondition(
      \Bs\Model\Childrens\Children::TABLE_NAME . ".id IN "
         . $this->db->pasteSubSelect($subselect)),
   "age", true, $having = "", $orderBy = "", $limitOffset = "",
   $limitRowCount = "",
   $this->db->pasteJoin(\Bs\Model\SchoolYear::TABLE_NAME,
      \Bs\Model\SchoolYear::TABLE_NAME . ".id = $schoolYearId")
);
```



Karta třídy Zajíčci pro školní rok 2011/2012

4leté děti (celkem 2)

Příjmení	Jméno	Datum narození	Matka	Otec	Integrace	Školné
Krula	Jan	26. 3. 2008	Krulová Hana	Krula Ivan	-	650 Kč
Škrletová	Magdalena	2. 2. 2008	Škrletová Alexandra	Škrleta Vladislav	0914a	650 Kč

5leté děti (celkem 2)

Příjmení	Jméno	Datum narození	Matka	Otec	Integrace	Školné
Koš	Jiří	6. 2. 2007	Košová Jitka	Koš Filip	0906	650 Kč
Wilhelmová	Růžena	24. 4. 2007	Wilhelmová Elena	Wilhelm Pavel	0906	650 Kč

6leté děti (celkem 2)

Příjmení	Jméno	Datum narození	Matka	Otec	Integrace	Školné
Gryčová	Larisa	9. 6. 2006	Gryčová Jana	Gryč Milan	0912	650 Kč
Macek	Miroslav	14. 1. 2006	Macková Marie	Macek Jan	0906	650 Kč

7leté děti (celkem 1)

Příjmení	Jméno	Datum narození	Matka	Otec	Integrace	Školné
Nechutná	Eva	17. 3. 2005	Nechutná Nikola	Nechutný Lubomír	-	650 Kč

Tisknout Zpět na kartu třídy

Obrázek 11 – tisková sestava třídy

5.6 Správa školních pobytů

Školní pobyt je definován následujícími údaji:

- název,
- místo,
- datum konání od-do,
- typ pobytu,
- školní rok, ve kterém se pobyt konal,
- seznam dětí, které se pobytu účastnily.

5.6.1 Správa typů školních pobytů

Kromě základní správy jednotlivých záznamů obsahuje modul také správu dynamického číselníku, uchovávající informace o typech školních pobytů. Princip práce s ním je obdobný jako u správy integrací, nebo jazyků zmíněný v kapitole 5.1.4.

5.7 Správa uživatelů

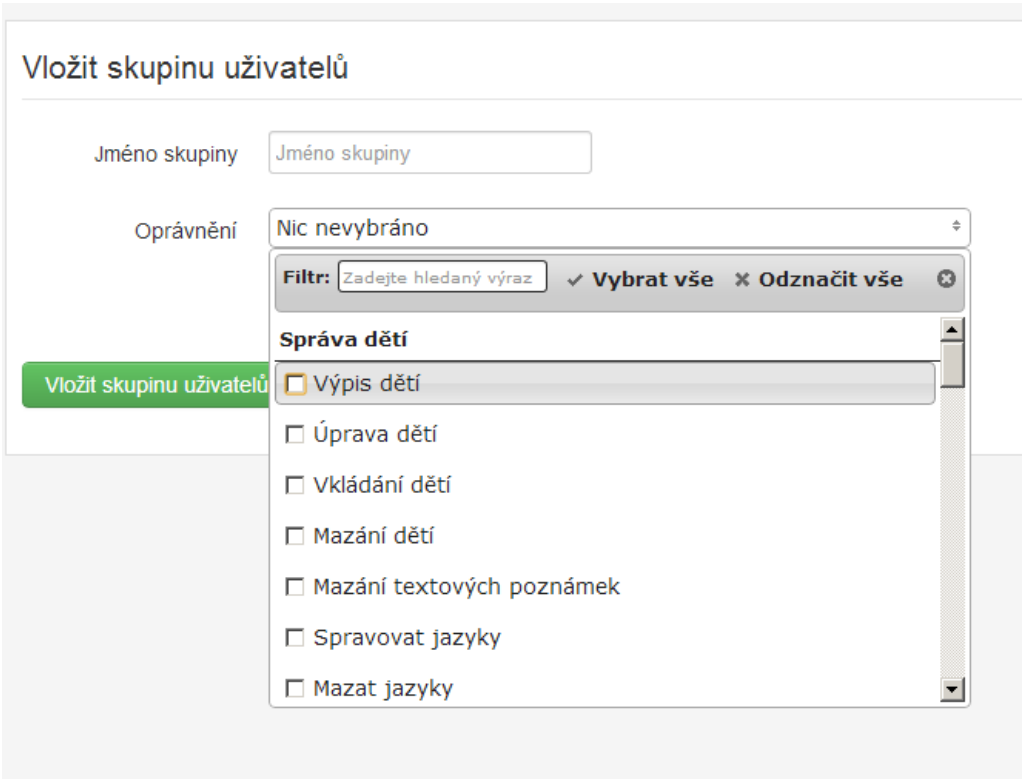
V aplikaci je implementováno řízení přístupu k jednotlivým akcím na základě členství uživatele ve skupině. Ve třídě `Authentication`, která je součástí registru, dojde po přihlášení

uživatele k načtení všech uživatelských práv. Ve třídě `AdminController` se při zpracování požadavku v metodě `checkAuthorization` kontroluje, jestli má uživatel dostatečná práva. Pokud tomu tak není, dojde k vyvolání výjimky `AuthorizationException`, která se následně odchytí a uživateli je vypsána chybová hláška.

Uživatelská práva se také kontrolují při sestavování uživatelského rozhraní a z grafického rozhraní jsou vypuštěny všechny prvky pro provádění akcí, na které nemá uživatel dostatečná práva.

5.7.1 Správa uživatelských skupin

Aby bylo možné přidělovat uživatelům nějaká práva pro práci se systémem, je nejdříve nutné založit skupinu uživatelů, u které se vyplní její název. Následně se z multiselectu vyberou všechny potřebná práva (Obrázek 12). Ty jsou seskupeny podle modulů aplikace.



Obrázek 12 – výběr oprávnění u skupiny uživatelů

5.7.2 Správa uživatelských účtů

U uživatelů systému potřebujeme udržovat následující údaje:

- přihlašovací jméno (musí být unikátní),
- příjmení,
- křestní jméno,
- tituly před a za jménem,

- datum narození,
- heslo,
- uživatelské skupiny, ve kterých je uživatel členem,
- příznak aktivní (pokud uživatel není aktivní, nemůže se přihlásit do systému, je však dále zachován v databázi a v budoucnosti je možné jej znovu aktivovat).

Uživatel může být současně členem více uživatelských skupin. Poté dojde ke sjednocení všech uživatelských práv.

Z důvodu zvýšení bezpečnosti jsou uživatelská hesla ukládána ve formě takzvaného salted hash. K původnímu heslu, které zadá uživatel, je připojen náhodně vygenerovaný řetězec – sůl. Na takto upravené heslo je následně použita hashovací funkce. Samotná sůl je ukládána jako další atribut tabulky `User`. Pokud by útočník chtěl použít pro získání hesla duhovou tabulky, musel by vytvořit všechny kombinace vstupních slov s danou solí – neví totiž na jakou pozici hesla je sůl vložena. Tato operace by byla velmi náročná a většinu útočníků odradí.

Úrovně jednotlivých oprávnění jsou dostatečně jemné na to, aby bylo možné vytvořit všechny potřebné role uživatelů. Je například možné vytvořit uživatele, který bude mít právo pouze na čtení záznamů. V praxi poté vzniknou 3 konkrétní role:

- Ředitelka – má plný přístup ke všem modulům aplikace, včetně správy uživatelů.
- Učitelka – také má plný přístup k systému, ale nemůže spravovat uživatele.
- Hospodářka – má přístup pouze k platebním údajům dětí.

5.7.3 Správa účtu přihlášeného uživatele

Uživatel, který nemá práva pro správu uživatelů, musí mít možnost měnit si vlastní údaje – po kliknutí na jméno pod kterým je přihlášen v levém panelu, se uživatel dostane na stránku s formulářem obsahující tyto údaje. Zde má možnost upravit všechny informace včetně změny hesla – to musí být pro kontrolu zadáno dvakrát. Jediný údaj, který není možné měnit je přihlašovací jméno.

5.8 Export a import dat

Aby bylo možné data ze systému jednoduše zálohovat, obsahuje aplikace modul pro export a import dat. Pokud má uživatel na tuto akci oprávnění, jsou mu v levém panelu zobrazeny ikony exportu a importu.

Po kliknutí na export dat, dojde automaticky ke stažení souboru s příponou sql, který obsahuje všechny data obsažené v databázi.

Před importem dat je uživatel nejdříve upozorněn, že po této akci dojde ke ztrátě všech změn, které byly provedeny od vzniku importovaného souboru. Aplikace kontroluje formát

vstupního souboru – pokud dojde k vybrání neplatného typu souboru, nebo souboru s poškozeným formátem dat, vypíše se uživateli chybová hláška.

Export a import dat je realizován funkcemi `exportDatabase` a `importDatabase` ve třídě `DatabaseGenerator`. Funkce pro export databáze využívá podobného algoritmu, jako skript pro generování struktury databáze. Postupně prochází všechny modelové třídy a získává z nich definici databázových tabulek. Z této definice poté zjistí všechny atributy tabulky pro příkaz `SELECT`. Data z vráceného dotazu parsuje do správné podoby příkazu `INSERT`. Tímto způsobem projde všechny tabulky a následně ještě relační tabulky. Vygenerované příkazy pro vložení dat do databáze jsou uživateli vráceny do souboru, jehož název obsahuje aktuální datum.

Opačný postup je mnohem jednodušší. V aplikaci již máme vygenerovanou strukturu databáze, stačí nám tedy načíst vstupní soubor a postupně provést všechny příkazy pro vložení dat ze souboru. Problém však nastává v tom, že při generování dat nikdy nezaručíme jejich správné pořadí. Může tedy nastat situace, že budeme vkládat záznam, který odkazuje na záznam jiné tabulky, ten ovšem ještě nebyl vytvořen.

Tento problém je vyřešen tak, že se vstupní SQL soubor rozdělí do pole po jednotlivých příkazech. Ty se postupně odebírají ze začátku pole a vykonávají se. Odchytávají se při tom vyvolané výjimky. Když se odchytí výjimka s příslušným chybovým kódem, dojde k zařazení vykonávaného příkazu na konec pole a pokračuje se dál v prováděném cyklu.

Před zahájením importu dat je nutné celou databázi zcela vyprázdnit. Může tedy dojít k situaci, kdy uživatel předá soubor s chybnou strukturou, aktuální data se z databáze odstraní a při importu dojde k vyvolání výjimky, která prováděné operace zastaví. V tuto chvíli by uživatel přišel o stará data a nebyl by schopen provést import.

Aby se předešlo této situaci, dochází ještě před smazáním databáze k jejímu exportu, jehož výstup je uložen do proměnné. Po odchycení výjimky vyvolané špatnou strukturou vstupního souboru dojde k importu ze starých dat a uživateli je vypsána chybová hláška.

```
while (!empty($sqlArray)) {
   $sqlQuery = array_shift($sqlArray);
   try {
      if (!empty($sqlQuery)) {
         $this->db->orderQuery($sqlQuery);
      }
   } catch (\Bs\CannotOrderQueryException $e) {
      if ($e->getCode() == 1452) {
         array_push($sqlArray, $sqlQuery);
      } else {
         $this->db->rollback();
         $this->importDatabase($oldData);
         throw $e;
      }
   }
}
```

6 Závěr

Po dvou semestrech pilné práce byla vytvořena aplikace, která plně vyhovuje všem současným potřebám mateřské školy. Aplikace je schopna uchovávat všechna nutná data, filtrovat v nich a vytvářet z nich různé typy tiskových sestav pro povinnou papírovou archivaci. Je vytvořena tak, aby byla uživatelsky přívětivá a intuitivní. V systému je možné vytvářet více uživatelů s různými uživatelskými právy. Program také nabízí možnost jednoduchého zálohování dat, což je pro dlouhodobý provoz důležité.

Nástroje vytvořeného frameworku plně dostačují tomuto typu aplikace a velmi dobré jsou i výsledky jeho výkonnosti. Rychlost generování stránky je zhruba srovnatelná s českým frameworkem Nette. Je ale nutné dodat, že Nette používá mocný nástroj v podobě cachování: tím docílí mnohem větší rychlosti při opětovném generování stránky. Do budoucna se tak naskýtá velmi zajímavá možnost vylepšení. Abych usnadnil budoucí práci s frameworkem, vytvořil jsem technickou dokumentaci, která popisuje jeho strukturu a API všech jeho modulů. Dokumentace je součástí přiloženého CD.

Aplikace byla vyvíjena ve spolupráci s mateřskou školou Klášterní v Liberci, kde od poloviny května běží v testovacím provozu. V nadcházejícím školním roce v ní budou evidovány všechny děti navštěvující školu. K aplikaci proto také vznikl uživatelský návod, který je součástí přiloženého CD.

Z hlediska softwarové architektury je systém navržen tak, aby ho do budoucna bylo možné jednoduše rozšířit, pokud by vznikl požadavek ze strany zákazníka. Vzhledem k použití internacionalizačního nástroje Gettext není žádný problém vytvořit libovolnou jazykovou mutaci aplikace.

Vzhledem k uchovávání osobních dat v databázi bylo nutné vyřešit bezpečnostní stránku aplikace. Všechny běžné typy útoků na webové aplikace plynoucí zejména ze špatně ošetřených vstupních polí formulářů řeší samotný framework. Programátorovi tedy odpadá zbytečná práce a zároveň se snižuje riziko na vznik neošetřeného vstupu. Při nasazení do produkční verze je použit šifrovaný protokol https, který zamezuje útoku typu man in the middle.

Seznam použité literatury

[1] GILMORE, W. *Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály*. 3. vyd. Překlad Jan Pokorný. Brno: Zoner Press, 2011, s. 171 Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.

[2] *Using Triggers* [online]. © 1997, 2013 [cit. 2013-05-15]. Dostupné z: http://dev.mysql.com/doc/refman/5.0/en/triggers.html

[3] *What is jQuery?* [online]. [2006] [cit. 2013-05-15]. Dostupné z: http://jquery.com/

[4] *Twitter Bootstrap: uživatelské prostředí webových aplikací hračkou* [online]. 5. 11. 2012 [cit. 2013-05-15]. Dostupné z: http://www.root.cz/zpravicky/twitter-bootstrap-uzivatelske-prostredi-webovych-aplikaci-hrackou/

[5] *About jQuery UI* [online]. [2007] [cit. 2013-05-15]. Dostupné z: http://jqueryui.com/about/

[6] PEACOCK, Michael. *Programujeme vlastní e-shop v PHP 5*. Vyd. 1. Brno: Computer Press, 2011, s. 34–42 ISBN 978-80-251-3181-7.

[7] *Cross Site Scripting* [online]. 23.11.2005 [cit. 2013-05-01]. Dostupné z: http://php.vrana.cz/cross-site-scripting.php

[8] *SQL Injection* [online]. 26.4.2013 [cit. 2013-05-01]. Dostupné z: http://www.php.net/manual/en/security.database.sql-injection.php

[9] VRÁNA, Jakub. *1001 tipů a triků pro PHP*. Vyd. 1. Brno: Computer Press, 2010, s. 362. ISBN 978-80-251-2940-1.

[10] VRÁNA, Jakub. *1001 tipů a triků pro PHP*. Vyd. 1. Brno: Computer Press, 2010, s. 367. ISBN 978-80-251-2940-1.

[11] *Man in the middle attack* [online]. 16.12.2001 [cit. 2013-05-01]. Dostupné z: http://www.krypta.cz/articles.php?ID=94

Přiložené CD

Přiložené CD obsahuje instalační distribuci, u které je umístěn návod na instalaci a spuštění aplikace. Dále se na disku nachází elektronická verze bakalářské práce, uživatelský návod na použití aplikace a technická dokumentace k vytvořenému frameworku.