Mendelova univerzita v Brně
Provozně ekonomická fakulta

Webový informační systém pro správu stromových školek

Bakalářská práce

Vedoucí práce:
Ing. Jiří Lýsek, Ph.D.

Tomáš Fiala

Brno 2015

Rád bych poděkoval svému vedoucímu Ing. Jiřímu Lýskovi, Ph.D. za přínosné konzultace při tvorbě bakalářské práce. Také děkuji rodině a přátelům za jejich psychickou podporu.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Webový informační systém pro správu stromových školek** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 22. května 2015 _____

Abstract

Fiala, T. Information system for administration of tree nurseries. Bachelor thesis. Brno: Mendel University, 2015.
This thesis describes design and implementation of information system for administration of tree nurseries. The second part is comparison between work in pure PHP and PHP framework.

Keywords

Information system, tree nursery, PHP framework, Symfony.

Abstrakt

Fiala, T. Webový informační systém pro správu stromových školek. Bakalářská práce. Brno: Mendelova univerzita v Brně, 2015.
Tato práce se zabývá návrhem a implementací webového informačního systému pro správu stromových školek. Druhou částí je porovnání práce v čistém PHP a PHP frameworku.

Klíčová slova

Informační systém, stromová školka, PHP Framework, Symfony.

Obsah

1 Úvod

Stromkové školky slouží k množení a pěstování stromků za účelem jejich následného prodeje. Firmy zabývající se tímto oborem mají zpravidla mnoho stromových školek rozmístěných v okolí sídla firmy. Evidence se často provádí papírově, ve formě tzv. plánů školek a také v tabulkovém procesoru. Vkládání dat na 2 místa je zbytečně časově náročné a přináší to prostor pro vytváření chyb.

Značným zjednodušením práce je evidence činností v informačním systému. Díky potřebě evidence různých specifických parametrů nelze použít univerzální systémy pro evidenci pracovních činností.

Návrhem a implementací takového systému se zabývá tato práce.

2 Cíl práce

Cílem práce bylo navrhnout a implementovat informační systém pro správu stromových školek. Systém bude kvůli použití frameworku jednoduše rozšiřitelný, proto by nebyl problém rozšířit jej na informační systém pro správu celé firmy. Systém implementuje hlavně část mezi vytvořením školky a vyoráváním vypěstovaných stromků.

3 Rešerše

Existuje mnoho prací s ovocnářskou tématikou, ty se však zabývají šlechtěním, pěstováním apod. Jedinou relevantní prací s podobnou tématikou je bakalářská práce Ondřeje Letochy (Letocha, 2009). Implementovaný informační systém je ovšem zaměřen hlavně na třídění a následný prodej stromků po jejich vyorání. Z tohoto důvodu není možné v systému evidovat jednotlivé práce, ani jejich statistiky. Systém má i další omezení, jakými jsou např. ukládání vazače a očkováka v databázi pouze jako řetězec, což znemožňuje uložení informací o více očkovácích v případě přeočkování a také způsobuje redundanci dat. Jelikož se jedná o starší práci, není zde použit ani žádný PHP framework, tudíž by náklady na údržbu a další rozšíření byly dost vysoké.

Druhou oblastí této práce jsou PHP frameworky, hlavně Framework Symfony 2. Tímto frameworkem se zabývá několik prací, jako například bakalářská práce Jana Dočkala, „Tvorba webových aplikací pomocí frameworku Symfony“ (Dočkal, 2015). Autor se věnuje porovnání frameworku Symfony s dalšími používanými frameworky. V práci jsou také uvedeny principy práce s frameworkem.

Knih zaměřených na frameworky je opravdu málo. Pro framework Nette nebyla vydána žádná. Codeigniter je na tom s literaturou lépe, díky stáří knih je ovšem jejich obsah poměrně zastaralý. Symfony nabízí na svých stránkách 4 knihy zdarma ke stažení, jejich obsah je vygenerovaný z velké části dokumentace pro každou verzi.

4 Základní pojmy

4.1 PHP

Skriptovací programovací jazyk PHP vytvořil dánsko-kanadský programátor Rasmus Lerdorf v roce 1994. Nejdříve vytvořil jednoduchý systém pro evidenci přístupů k jeho stránkám v Perlu, následně jej musel, z důvodu vysokého vytěžování serveru, přepsat do jazyka C. Původně tento systém sloužil k jeho osobnímu použití, ostatním uživatelům serveru se ovšem také zalíbil a proto jej následně uvolnil pod názvem „Personal Home Page Tools“.

Jedním z dalších projektů pana Lerdorfa byl nástroj pro používání SQL dotazů na stránkách, vytváření formulářů a zobrazování výsledků dotazů s názvem „Form Interpreter“, tyto dva projekty byly následně sloučeny do jednoho s názvem PHP/FI 2.0. Od verze 3.0 se používá opět pouze zkratka PHP, která ovšem znamená hypertextový preprocesor PHP (PHP: Hyper-text Preprocessor). (Kosek, 1999)

Jednotlivé části jazyka se postupně vyvíjely, jako například Zend jádro v PHP 4.0, nebo postupné zavádění OOP od PHP 5.1, až dospěly k dnes používané verzi PHP 5.6. Souběžně s PHP 5 se také vyvíjela verze 6 podporující plnohodnotné Unicode kódování. Verze 6 ovšem kvůli problémům s implementací Unicode nebylo vydáno, a další vydanou verzí bude PHP 7 plánované na druhou polovinu roku 2015.

Jazykem PHP se zabývá okolo 400 anglicky psaných knih (PHP: Books about PHP, 2014). Během vytváření bakalářské práce budou používány knihy, které se zabývají pokročilejšími technikami programování v PHP, jimiž jsou PHP - objektově orientované: koncepty, techniky a kód (Lavin, 2009) a PHP 6 : programujeme profesionálně (LECKY-THOMPSON, NOWICKI, 2010)

Jazyk PHP je v dnešní době nejpoužívanější jazyk pro webové stránky s 82 % podílem (Usage of server-side programming languages for websites, 2015), napomáhá tomu hlavně jeho velice "strmá" křivka učení. Díky jeho oblíbenosti je podporován na převážné většině webhostingů, což se nedá říct o konkurenčních jazycích, kterými jsou např. ASP.NET, nebo Ruby.

Každý PHP skript začíná znaky `<?php` a (volitelně) končí `?>`. K jeho interpretaci je potřebný webový server nebo jej lze spouštět přes příkazovou řádku. Syntaxe je velmi podobná většině dnes používaným jazykům, jako například C nebo Java. Hlavním rozdílem je používání znaku $ před názvem proměnné (jako v jazyce Perl) a velmi slabá typová kontrola.

4.2 HTML a CSS

HTML je sjednocující značkovací jazyk pro WWW. Tento jazyk vznikl v roce 1991 na základě dokumentu jménem „HTML Tags“ napsaného sirem Timem Berners-Leem, který obsahoval necelé 2 tucty značek (neboli tagů). HTML je ovšem založeno na mnohem starším jazyce SGML. Vývoj jazyka prošel několika důležitými fá-

zemi. 1. specifikací bylo HTML 2.0, které podporoval tehdy nejrozšířenější prohlížeč „Mosaic“. Následně vznikly 2 větve vývoje, XHTML povolující pouze validní XML značky (např. ukončování nepárových značek lomítkem a zápis značek malými písmeny) a HTML, jež nebylo tak striktní.

Nyní se používá (zatím nedokončený) standard HTML5, který tyto větve sjednotil a zavedl poměrně velké množství inovací. HTML5 přinesl nové tagy pro rozvržení stránky, kterými jsou `article`, `section`, `header` a `footer`, proto se nyní nemusí tyto části řešit pomocí tagu `div`, to přináší hlavně lepší zpracování dat roboty (Alexis Goldstein, Louis Lazaris, 2011). Další novinkou jsou tagy pro práci s multimediálním obsahem (audio a video), které časem nahradí dnes čím dál méně podporovaný Adobe Flash v oblasti přehrávání audia a videa. Nově bylo zavedeno také zjednodušení některých zbytečně složitých zápisů, jako například deklarace typu dokumentu, pro kterou se v předchozích standardech používalo (pro XHTML 1.0 strict) `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">` se dnes používá pouze `<!doctype html>`. Také některé atributy nejsou již potřebné, jelikož se zde dnes už téměř nepoužívají jiné jazyky, než Javascript pro interaktivní prvky a CSS pro formátování stránky. (Keith, 2010)

Dnes již prakticky nemá význam používat knihy zabývající se staršími standardy HTML a CSS. Knihy HTML5 for web designers (Keith, 2010) a CSS3 for web designers (Cederholm, 2010) se zabývají jejich tématy stručně a s mnoha praktickými příklady. Další knihou je HTML5 a CSS3: Výukový kurz webového vývojáře (Hogan, 2011), která prezentuje tyto technologie na konkrétních projektech. Tyto knihy jsou psané pro pokročilejší programátory, proto je potřeba alespoň trochu ovládat HTML a CSS.

4.3 Javascript a jQuery

Javascript je interpretovaný programovací jazyk, který byl vytvořen firmou Netscape a standardizován v roce 1997 (A Short History of JavaScript, 2012). Jedná se o jazyk převážně interpretovaný na klientské straně. Jeho hlavním úkolem je interaktivita webových stránek. Používá se také pro kontrolu dat formulářů na klientské straně před jejich odesláním. Jelikož jsou některé funkce rozdílně implementovány v různých prohlížečích, vzniklo mnoho knihoven, které tyto rozdíly eliminují. Tato nekompatibilita mezi prohlížeči je hlavním důvodem, proč se místo čistého Javascriptu dnes nejčastěji používá knihovna jQuery.

Hlavní výhoda jQuery je, mimo jeho kompatibility mezi prohlížeči, výběr prvků za pomoci CSS selektorů a následné spuštění kódu podle odchycené události (kterou může být např. kliknutí na prvek). U odchycené události se často spouští bezejmenná funkce, pomocí tzv callbacku, to znamená, že je funkce poslána jako parametr jiné funkce. Tento zápis je o mnoho jednodušší na údržbu a čitelnější, než v případě čistého Javascriptu, kde musela být volaná funkce definována u každého elementu zvlášť. Knihovna jQuery také nabízí výrazné zjednodušení práce s AJAXem díky několika zabudovaným funkcím. Je zde také mnoho funkcí pro práci

s animacemi a zobrazování/skrývání prvků. Dnes je však možné na velkou část animací používat CSS3.

Pro odlaďování javascriptového kódu se používají, ve webových prohlížečích, různé vývojářské nástroje. Chrome má již v základu „DevTools“, umožňující zobrazení html prvků, síťového provozu, nastavení breakpointů a mnoho dalších užitečných funkcí. V jedné z posledních verzí byla také zavedena také podpora testování responsivního designu pomocí „emulace“ různých zařízení. Vývojářské nástroje v prohlížeči Internet Explorer jsou vhodné hlavně k testování zobrazení v jeho starších verzích. V prohlížeci Mozilla Firefox dříve bylo, kvůli rozšíření funkcí, nejlepší doinstalovat doplněk Firebug, který nabízel podobné vlastnosti jako vývojářské nástroje prohlížeče Chrome. Dnes vývojářské nástroje Firefoxu (pojmenované Aurora) obsahují vše důležité a žádný doplněk již není potřeba.

4.4 PHP frameworky

PHP frameworky mohou používat několik návrhových vzorů, nejpoužívanější jsou ovšem MVC PHP frameworky, rozdělující webovou aplikaci na 3 základní vrstvy – model (práce s databází), view (zobrazování dat) a controller (zpracování dat z modelu a jejich odeslání do view a opačně). Tyto frameworky také poskytují mnoho nástrojů pro vytváření webových aplikací, díky nimž je kód snadněji udržovatelný, obsahuje méně chyb a v ideálním případě i přenositelnější na další projekty. Hlavním přínosem PHP frameworků je použití často využívaných obecných funkcí pro opakující se činnosti u více projektů. Proto nemusí programátoři psát znova téměř stejnou část kódu.

Z důvodu implementace informačního systému ve frameworku Symfony bude tomuto frameworku věnována větší pozornost.

4.4.1 Nette Framework

Nejpoužívanějším MVC frameworkem je v ČR Nette Framework od českého vývojáře Davida Grudla. Framework Nette dříve trpěl bouřlivým vývojem a spousta funkcí nebyla zpětně kompatibilních, dnes se již vývoj ustálil a zaměřuje se hlavně na zdokonalování jednotlivých částí. Proto existuje na internetu mnoho tutoriálů, které jsou již neaktuální a k seznámení se s jazykem nepoužitelné, viz seriál „Začínáme s Nette Framework“ (Grudl, 2009). (Tögl, 2012)

David Grudl je hlavním, a téměř jediným programátorem tohoto frameworku, proto je jeho budoucnost poměrně nejistá oproti celosvětově rozšířeným frameworkům, které mají velkou komunitu programátorů. Tato skutečnost je hlavním důvodem, proč nebyl framewok Nette vybrán pro implementaci této bakalářské práce.

Dokumentace Nette (Nette Framework, 2014) obsahuje i přes její neustálé vylepšování spíše základy jednotlivých částí a proto je poměrně často potřeba hledat na oficiálním fóru. Naštěstí je komunita vývojářů v ČR poměrně široká a o mnohých věcech již bylo na těchto fórech diskutováno. Popisem Nette se zabývá i něko-

lik diplomových a bakalářských prací, bylo by ovšem mnohem přívětivější pokud by byla vydána oficiální kniha zabývající se frameworkem podrobněji.

4.4.2 Symfony

Dalším z populárních frameworků je Symfony, jeho historie sahá do roku 2005, kdy byl Fabienem Potencierem vytvořen „Sensio Framework“ pro firemní účely. Ten byl o několik měsíců později uvolněn s open-source licencí pod názvem Symfony. V roce 2011 byl Framework z velké části přepsán a vznikl Symfony 2. Dnes je aktuální verze 2.6 a verze 2.3 s dlouhodobou podporou.

Framework je složen z přibližně dvaceti nezávislých knihoven, z nichž může být každá použita zvlášť v jakémkoliv projektu. Tyto knihovny poskytují různorodé nástroje vhodné tvorbu webových aplikací:

- „HttpFoundation“ sloužící pro práci s http požadavky a odpověďmi
- „Form“ pro jednoduché vytváření formulářů a jejich odesílání
- „Validator“ k validaci vstupů dle určitých pravidel
- „ClassLoader“ umožňuje tzv. „autoloading“, což zajišťuje automatické načítání PHP tříd dle jejich názvu a jmenného prostoru
- „Translation“ - nástroje pro překlad aplikace do různých jazykových mutací
- a mnoho dalších

Tento soubor nástrojů umožňuje rapidní zrychlení vývoje aplikací a zajištění jejich bezpečnosti. (Symphony book, 2014)

Nejlepším zdrojem pro prozkoumání a naučení tohoto frameworku se jeví oficiální dokumentace. V ní je, mimo jiné, možné objevit 4 základní knihy. První knihou je The Book (Symphony book, 2014), která obsahuje všechny základní informace pro vývoj webových aplikací. Druhou, pokročilejší knihou je The Cookbook (The Cookbook, 2014) obsahující mnoho praktických příkladů jak správně řešit různé situace ve frameworku. Třetí knihou je „The Components“ (The Components,2014), ta popisuje jednotlivé komponenty frameworku. A poslední, čtvrtou knihou je „Symfony Best Practices“ Symfony Best Practices, 2014) s popisem správných postupů při implementaci jednotlivých částí aplikace v Symfony.

Všechny komponenty Symfony jsou řešeny jako tzv. bundle (česky balík). Každá webová aplikace musí být také řešena jako jeden, nebo více bundlů. Jednotlivé bundly je možné používat i v dalších aplikacích. Díky této modularitě je velice jednoduché stáhnout bundle třetí strany a používat jej v aplikaci.

Věcí, na kterou je potřeba si zvyknout u Symfony je adresářová struktura. Kvůli dělení aplikace na bundly je tato struktura, zejména pro začínající programátory v tomto frameworku, poměrně matoucí. Vytvořená aplikace je ve složce src/název bundle. Ta se dělí na controller (se všemi controllery), src (obsahující složku view se šablonami rozdělenými do složek podle controllerů), config (s nastaveními v rámci bundle) a další.

Důležitou součásti tohoto frameworku je ovládání pomocí konzole, která dokáže ušetřit mnoho času vygenerováním obsahu některých souborů. Vytváření nového controlleru se provádí přes příkaz `php app/console genera-`

`te:controller`. Tímto příkazem se spustí interaktivní vytváření controlleru, po vyplnění názvu controlleru, výběrem typu anotací a volitelném nadefinování názvů akcí se vygenerují všechny potřebné soubory, včetně souborů šablon pojmenovaných podle nadefinovaných akcí. Druhým způsobem generování je neinteraktivní cesta, kde se všechny údaje zadávají jako parametry příkazové řádky. Podobně lze generovat i mnoho dalších částí systému, jak je vidět na obr. č. 1.

```
doctrine
  doctrine:cache:clear-metadata         Clears all metadata cache for an entity manager
  doctrine:cache:clear-query            Clears all query cache for an entity manager
  doctrine:cache:clear-result           Clears result cache for an entity manager
  doctrine:database:create              Creates the configured databases
  doctrine:database:drop                Drops the configured databases
  doctrine:ensure-production-settings   Verify that Doctrine is properly configured for a production environment.
  doctrine:generate:crud                Generates a CRUD based on a Doctrine entity
  doctrine:generate:entities            Generates entity classes and method stubs from your mapping information
  doctrine:generate:entity              Generates a new Doctrine entity inside a bundle
  doctrine:generate:form                Generates a form type class based on a Doctrine entity
  doctrine:mapping:convert              Convert mapping information between supported formats.
  doctrine:mapping:import               Imports mapping information from an existing database
  doctrine:mapping:info                 Shows basic information about all mapped entities
  doctrine:query:dql                    Executes arbitrary DQL directly from the command line.
  doctrine:query:sql                    Executes arbitrary SQL directly from the command line.
  doctrine:schema:create                Executes (or dumps) the SQL needed to generate the database schema
  doctrine:schema:drop                  Executes (or dumps) the SQL needed to drop the current database schema
  doctrine:schema:update                Executes (or dumps) the SQL needed to update the database schema to match th
e current mapping metadata
  doctrine:schema:validate              Validates the doctrine mapping files
generate
  generate:bundle                       Generates a bundle
  generate:controller                   Generates a controller
  generate:doctrine:crud                Generates a CRUD based on a Doctrine entity
  generate:doctrine:entities            Generates entity classes and method stubs from your mapping information
  generate:doctrine:entity              Generates a new Doctrine entity inside a bundle
  generate:doctrine:form                Generates a form type class based on a Doctrine entity
init
  init:acl                              Mounts ACL tables in the database
jordillonch
  jordillonch:generate:crud             A CRUD generator with paginating and filters.
```

Obr. 1 Příkazy konzole symfony

5 Porovnání čistého PHP a práce s PHP frameworkem

Informační systémy, a obecně jakoukoliv webovou aplikaci, je možné programovat pomocí čistého PHP nebo s využitím některého z frameworků. Použití čistého PHP má oproti práci s frameworky ovšem své klady a zápory.

Hlavní nevýhodou frameworků je oproti PHP rychlost a paměťová náročnost, tyto neduhy ovšem vynahrazují mnohými "vymoženostmi", které zajišťují vyšší úroveň zabezpečení a hlavně zvýšení efektivity práce. PHP frameworků existuje velké množství a z nichž každý nabízí různé úrovně abstrakce, šablonovací systémy, atd. Téměř všechny z nich však používají časem prověřený návrhový model MVC (nebo velmi podobný MVP), kde je oddělena práce s databází, zpracování dat a jejich výpis.

Tato část se zabývá porovnáním frameworků s prací v čistém PHP. Jsou zde uvedeny obecné informace a konkrétní případy jsou demonstrovány na frameworku Symfony.

5.1 MVC architektura

Velkým problémem PHP je, že není přesně definován způsob zápisu aplikací. Proto je možné udělat celou aplikaci v jednom PHP skriptu. Tento způsob programování nevadí u velice malých aplikací, zejména kvůli rychlosti naprogramování takové aplikace. V praxi je ovšem nepoužitelný kvůli složité rozšiřitelnosti, problémovém hledání chyb v kódu a téměř nulové znovupoužitelnosti.

Proto většina dnešních PHP frameworků používá architekturu MVC (Model, View, Controller). Tato architektura rozděluje aplikaci do tří úrovní abstrakce.

Nejnižší úrovní je práce s databází, nebo jiným úložištěm dat, kterou zajišťuje "Model". Ten získává z databáze potřebná data a také zprostředkovává jejich ukládání, editaci a mazání.

Další úrovní je zpracování dat za pomoci controlleru. Ten se stará o komunikaci mezi pohledem (View) a modelem. Přijatá data z modelu zpracuje a posílá do šablony. Stejným způsobem funguje posílání dat druhým směrem, kde se přijatá formulářová data posílají do modelu pro jejich následné uložení do datového úložiště. Controller se stará i o všechny transformace dat a kontrolu jejich správnosti. Veškerá logika aplikace se nachází také zde.

Data, která controller zpracuje, posílá do šablony (View) a s její pomocí se vykreslí do prohlížeče. View je často řešen pomocí různých šablonovacích systémů, kvůli optimalizaci může být také řešen přes čisté PHP.

5.2 Object-relation-mapping

Hlavně pro zajištění přenositelnosti mezi více databázemi se používá Object-relation-mapping (dále jen ORM). ORM funguje na principu entit, podle kterých se následně generuje databáze.

Každou entitou se rozumí objekt, jehož atributy představují jednotlivé sloupce v databázi. Atributy třídy jsou vždy soukromé a přistupuje se k nim přes pomocnou metodu, která může kontrolovat typ vstupních dat a v případě cizího klíče nastavit atributy u vázané entity.

S ORM entitami se následně pracuje přes „query builder“, u nějž je možné filtrovat data i bez znalosti SQL. Query builder obsahuje intuitivně pojmenované funkce (select, where, join) a jejich parametrem může být asociativní pole. Doctrine umožňuje jako alternativu ke „query builderu“ používat DQL (Doctrine query language), který je velice podobný jazyku SQL. Rozdílem je zjednodušení práce s vazbami a uvádění názvů entit místo názvů tabulek. Poslední možností je pracovat s čistými SQL dotazy, to ovšem není doporučováno kvůli různým optimalizacím v „query builderu“.

5.3 Zabezpečení

Při programování v čistém PHP je nutné vždy ošetřovat všechny vstupy, aby se zamezilo různým útokům. Oproti tomu frameworky mají součásti řešící ochranu proti těmto útokům, a programátor se tedy většinou těmito problémy nezabývá, nebo používá funkce zahrnuté ve frameworku pro tyto účely.

5.4 Pomocné funkce

PHP frameworky obsahují obrovské množství funkcí, které usnadní programátorovi práci. Také řeší činnosti, které se používají téměř na všech projektech, jako přihlašování uživatelů a správa jejich oprávnění.

V controlleru se jedná zejména o funkce pro zpracování řetězců (a hlavně ošetření dat pro vložení do databáze), práci s obrázky, odesílání e-mailů a mnoho dalších, které již není potřeba opakovaně vytvářet.

Šablony mají funkce pro formátování výstupu, které se hodí například při zobrazování desetinných čísel, u kterých se v ČR musí používat desetinná čárka místo desetinné tečky používané pro uložení hodnoty v databázi.

5.5 Šablonovací systémy

Velkým zjednodušením práce pro kodéry byl vznik šablonovacích systémů. Do šablony se posílají již zpracovaná data, a proto je možné se zaměřit pouze na jejich správné zobrazení. Stejná data z controlleru můžou být zaslána do několika šablon, např. při exportování do různých souborů (csv, xml). Výhodou šablonovacích sys-

témů je zjednodušení práce v týmu, kde pracují programátor a kodér v jiných souborech a proto nehrozí neúmyslné přepsání souboru.

Každý z těchto systémů má svůj vlastní způsob zápisu, což sice může být menší problém při přechodu z jednoho na druhý, oproti šablonám psaných v PHP je ovšem zajištěna mnohem lepší čitelnost kódu. Tato čitelnost je vykoupena větší náročností, protože musí být šablona přeložena do PHP kódu.

Asi nejznámějšími šablonovacími systémy jsou dnes Twig (používaný ve frameworku Symfony), Latte (Nette), Smarty a Blade (Laravel).

5.6 Zpracování HTTP požadavků

Při programování jakékoli webové aplikace se žádný programátor neobejde bez formulářů a následného zpracování dat z nich odeslaných.

V PHP jsou data posílána přes superglobální proměnné $_GET a $_POST, popř. $_REQUEST. Samotné PHP ovšem nemá v základu žádnou kontrolu nad tímto obsahem a tudíž ani ochranu proti různým útokům. Proto se musí při každém ukládání a zobrazování dat do databáze provádět kontrola.

Symfony řeší zpracovávání dat v controlleru za pomoci proměnné typu Request a jeho funkcí `getRequest()->query->all()` získá všechny data. Ať už slouží data pro filtrování dat z databáze, nebo jejich ukládání, SQL injection je ošetřeno za pomoci ORM Doctrine.

Pro ještě jednodušší práci je možné používat formuláře naimplementované v Symfony, které se nadefinují v controlleru a v šabloně se pomocí 3 příkazů (`form_start, form_widget, form_end`) vykreslí. Pro zpracování a uložení dat se často používá stejná funkce, protože obsahuje kontrolu správnosti dat ve formuláři.

5.7 Odlaďování aplikace

Pro odlaďování aplikace obsahuje každý Framework nástroj zvaný „profiler“. Profilery se v testovacím prostředí zobrazují ve formě lišty na spodní části obrazovky. Po jejich otevření se zobrazí stránka s podrobnějšími údaji.

Každý profiler obsahuje užitečné nástroje pro ladění aplikace. Je v něm často zobrazen aktuálně používaný controller a jeho akce, délka trvání zpracování skriptu, velikost použité paměti, přihlášený uživatel a počet dotazů na databázi. Zobrazované údaje se mohou v různých frameworcích lišit, v Symfony se k těmto hodnotám ještě zobrazí problémy s entitami, verze PHP a verze Symfony. Při otevření profileru se zobrazí podrobné údaje s průběhem zpracování stránky, těmi mohou být např. provedené dotazy na databázi, výpis přijatých dat, a další. Profilery také umožňují analýzu trvání jednotlivých volání funkcí, z té je možné hned vidět část kódu, která zpomaluje aplikaci.

Při chybě se vypíše popis chyby a soupis volaných funkcí s jejich parametry. V případě čistého PHP by se zobrazil pouze řádek s chybou a odlaďování by bylo složitější.

Na produkčním serveru se nepoužívá testovací prostředí a při chybných stavech aplikace se uživateli zobrazí chyba č. 500 (Internal server error) a chyba se uloží do logu.

5.8 Optimalizace

Vysokou paměťovou a časovou náročnost částečně vynahrazují frameworky využitím cache. Tu je možné používat i v projektech s čistým PHP, její zprovoznění ovšem stojí programátora zbytečný čas, který je možné investovat do vylepšování aplikace.

5.9 Zhodnocení

V dnešní době se při větších projektech používat frameworky jistě vyplatí. Programátoři se můžou soustředit na samotnou aplikaci a ne na jádro systému. Proto lze s využitím frameworku rychle vytvořit spolehlivou a bezpečnou aplikaci s možností jednoduché rozšiřitelnosti.

6 Nároky na IS pro správu stromových školek

Existuje několik druhů školek, které se liší svým účelem využití nebo druhem rostlin. Tato práce je zaměřena hlavně na evidování činností prováděných na ovocných školkách, vzniklý informační systém by ovšem mohl být používán i u jiných typů, některé jeho části by ovšem musely být mírně upraveny.

V této části budou přiblíženy některé činnosti prováděné na ovocných školkách. Systém zahrnuje evidenci všech těchto činností včetně statistik.

6.1 Výběr vhodného umístění školky

Vhodné místo pro školku je potřeba vybrat podle aktuálně nabízených pozemků k pronájmu, či prodeji. Tato činnost se bude stále muset dělat ručně, protože neexistuje žádný centralizovaný systém s možností exportu dat o nabízených pozemcích. Pronájmy a prodeje obecních pozemků bývají většinou zveřejňovány na úředních deskách jednotlivých obcí.

Pro správný růst stromků ve školce se musí vybírat takové pozemky, které vyhovují pěstovaným stromkům. U ovocných školek je nutné zajistit pozemky s nadmořskou výškou v rozmezí 250 - 400 m n. m., nezamořenou půdou, nejlépe s dřívějším pěstováním pouze polních rostlin, a dostupným zdrojem nezávadné vody.

6.2 Příprava půdy pro školku, založení školky

Před samotným založením školky se musí provést hluboká orba a případně zajistit úprava půdy k vytvoření ideálních podmínek pro růst rostlin. Následuje oplocení a vysazování podnoží.

6.3 Štěpování

Štěpování lze rozdělit na 2 části – roubování a očkování (forkartování, nověji čipování).

6.3.1 Roubování

Roubování je způsob rozmnožování ovocných dřevin, který se provádí v době vegetačního klidu. Rozděluje se na roubování v ruce, nebo na poli.

Roubování v ruce se většinou provádí na prostokořenné podnože v hale. Nejčastější způsob roubování je kopulace, nebo anglická kopulace. Tento způsob se provádí nejčastěji během ledna a února. Takto naroubované podnože je potřeba uložit do chladírny. Výsadba nastává během března a dubna, Během výsadby je již roub částečně zakalusen a rašit začíná až po zaškolkování. Výhodou tohoto způsobu roubování je, že se může provádět při nevhodném počasí, jelikož se provádí ve vytápěné místnosti.

Roubování na poli se provádí až během března přímo ve školce, na podnože vyškolkované v minulém roce. Takto naroubované podnože ihned začínají rašit.

6.3.2 Očkování

Očkování se provádí více způsoby: Očkování do T (pod kůru), Forkertování (novější název je čipování). Provádí se za první a za druhé mízy.

Očkování za druhé mízy se provádí během srpna, v době kdy stromek začíná stahovat mízu do kořenů na vyčištěnou část podnože (kmínek), buď klasickým způsobem pod kůru do T, nebo forkertováním (očko obsahuje i dřevěnou část roubu). Takto přenesené očka již v roce očkování neraší, rašení nastává až v dalším roce.

Očkování za první mízy provádíme během dubna pouze forkertovým způsobem, T způsob není možný, jelikož používáme rouby, které byly zachlazeny a nemají mízu, tudíž by nebylo možné oddělit kůru od dřeva. Hlavní výhodou tohoto způsobu je možnost přeočkování podnoží, které byly očkovány v srpnu, a očko se neujmulo, proto můžeme tento způsob nazývat přeočkování.

6.4 Počítání stromků/podnoží

Pro kontrolu je potřeba také průběžně počítat stromky v jednotlivých řádcích. Počty těchto stromků se postupem času mohou hodně měnit, jelikož se poměrně často neujme část podnoží, roubů, nebo oček.

Počítání také slouží ke kontrole předešlých činností, hlavně u prací prováděných s úkolovou mzdou, kde by mohlo dojít úmyslnému navyšování počtu stromků na řádku.

6.5 Návrh systému

Po zvážení všech prováděných činnosti na ovocné školce bylo nutné navrhnout systém pro jejich evidenci. Součástí systému také musí být evidence školek a podrobností o stromcích v nich obsažených.

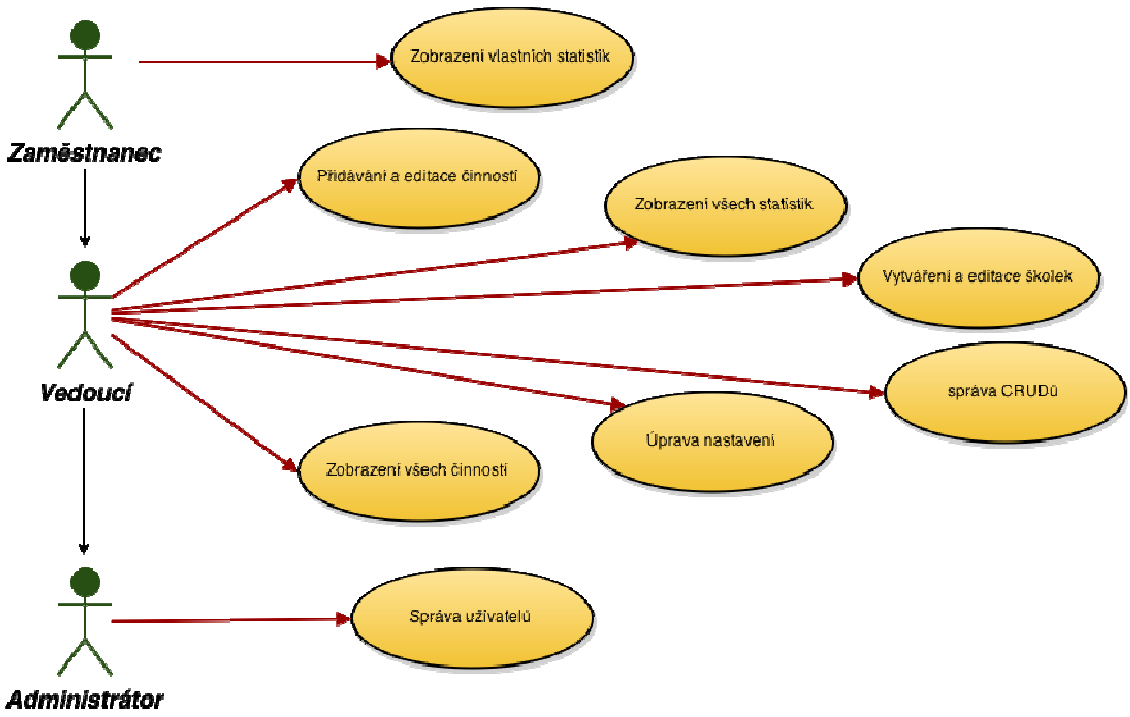
Návrh je proveden popisem požadavků a následným vytvořením use-case diagramu a Entity-relation diagramu.

6.5.1 Požadavky na systém

Informační systém pro správu stromových školek musí umožňovat evidenci pracovních činností se všemi podrobnostmi. U každého řádku (nebo jeho části) se musí ukládat informace o jednotlivých částech stromku. Většinou se na řádcích pracuje v týmu. Jednotlivé týmy mohou provádět na řádcích různé činnosti. U těch je nutné evidovat část řádku, nebo alespoň školku, a čas začátku a konce činnosti. Volitelně lze evidovat počet kusů, stroj a přidanou část stromku (roub nebo očko). Je nutné evidovat ztráty vzniknuté neuchycením částí stromku.

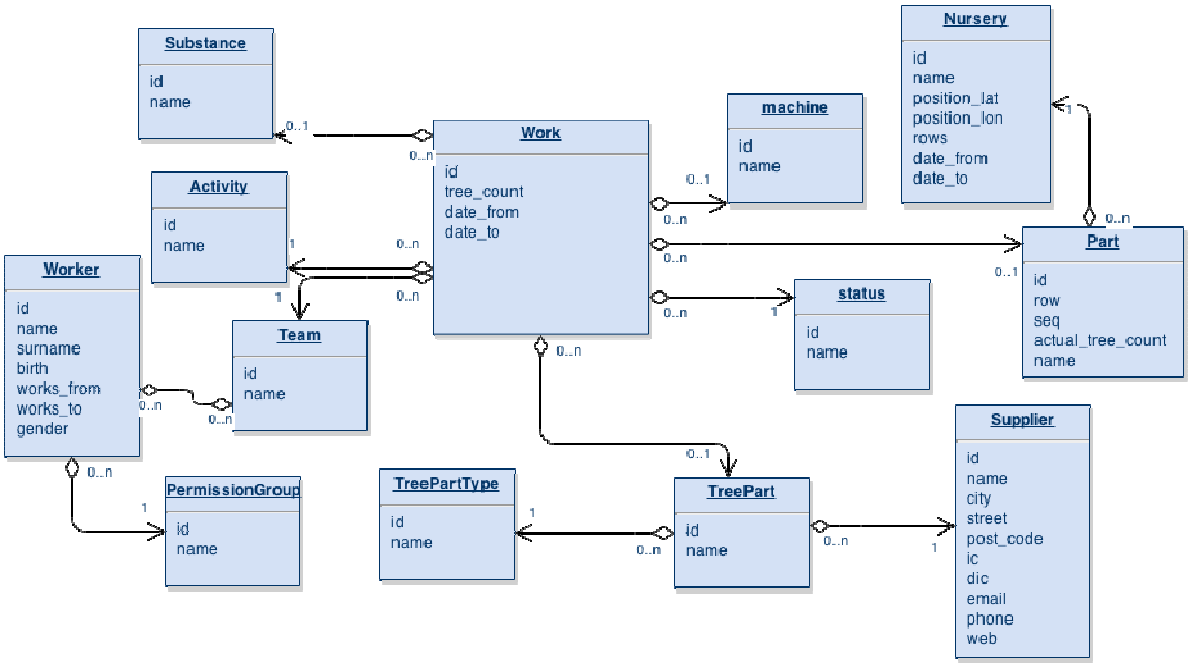
Uživatelé musí mít různé typy oprávnění, které znemožní přístup do některých částí systému. Administrátor má přístup do celého systému bez omezení. Vedoucí školky spravuje jednotlivé činnosti prováděné na školce. Každý zaměstnanec si může zobrazit svoje činnosti a statistiky.

6.5.2 Use-case diagram



Obr. 2 Use-case diagram navrhnutého systému

6.5.3 Entity-relation diagram



Obr. 3 Entity-relation diagram

7 Implementovaný IS

Po vyhodnocení všech požadavků na systém bylo potřeba navrhnutý informační systém naimplementovat. Během implementace byl brán ohled také na UX (user experience) a systém je tedy maximálně uživatelsky přívětivý.

7.1 Použité technologie

K implementaci IS byl použit PHP framework Symfony, který obsahuje ORM Doctrine. Data jsou uložena v MySQL databázi, ale kvůli použití Doctrine by nebyl problém tuto databázi nahradit jakoukoliv jinou bez nutnosti měnit kód aplikace. Mezi další použité technologie patří jQuery, CSS3, HTML5 a front-end framework Bootstrap. V některých částech jsou zobrazovány údaje na mapách, ty jsou řešeny pomocí Google Maps API.

7.2 Návrh databáze

První důležitou činností bylo navržení databáze. Jelikož tento systém používá ORM Doctrine, je databáze řešena popisem pomocí entit, z nichž každá reprezentuje tabulku v databázi. Jak je vidět z ERD na obrázku č. 3, databáze obsahuje 13 tabulek.

Pro návrh databáze byl použit program ORM designer, který nabízí jednoduché grafické rozhraní pro návrh Doctrine entit projektu.

Z navržených entit byly následně, za pomoci příkazu "`php app/console doctrine:schema:update --force`", vygenerovány všechny tabulky databáze včetně všech jejich atributů a cizích klíčů.

7.3 Postup implementace

7.3.1 CRUDy

Po vytvoření entit bylo potřeba vygenerovat "CRUDy" pro některé entity. Pro tuto činnost byl vybrán "bundle" jordillonch-crud-generator. Tento bundle poskytuje generování CRUDů včetně některých filtrů a se stránkováním. Další jeho vlastností je použití front-end frameworku Bootstrap 2. Vygenerované soubory ovšem nejsou dokonalé, proto bylo potřeba jejich velkou část přepsat, aby byly funkční s aktuální verzí frameworku bootstrap (což je verze 3) a doplnit chybějící filtry pro vazby s jinými entitami, které tento bundle nepodporuje.

U každé školky jsou evidovány její zeměpisné souřadnice, pro práci s nimi bylo použito Google maps API. Tyto souřadnice umožňují jednoduchý výběr školky pomocí mapy.

7.3.2 Operace

Operace používají z velké části AJAX. Ten sice v Symfony není řešen tak elegantně jako v Nette (tzn. pomocí "snippetů"), dá se ovšem jednoduše používat přes jQuery funkci "load", což je zjednodušení funkce "ajax", která odešle požadavek s daty na danou adresu a výsledek vypíše do prvku, u kterého se volá.

Grafické zobrazení školky je řešeno pomocí „glyphicons“, které jsou součástí frameworku Bootstrap. Glyphicons je sada různých monochromatických ikon a symbolů. V Bootstrapu jsou implementovány jako speciální font, každý symbol se tedy chová jako obyčejný znak. Do šablony se posílají entity částí řádku, ze kterých se získá počet stromků a v poměru k nejdelšímu řádku se zobrazí požadovaný počet symbolů. Jednotlivé části jsou na řádku odděleny symbolem tužky (který je vizuálně podobný v praxi používaným dřevěným kolíkům pro oddělení částí řádků). Vybraným částem je jednoduše změněna barva pomocí CSS. Pokud by byly pro symboly použity obrázky, tak by se musely měnit pro každou barvu, nebo používat CSS sprite (každý obrázek by obsahoval ikony s několika barvami).

Po vybrání částí řádků v grafické části se pomocí AJAXu vytvoří formulář pro vkládání nových operací. Parametrem tohoto ajaxového volání jsou vybrané řádky. Volaný skript následně vygeneruje formulář se společnými parametry pro všechny části řádku, kterými jsou tým, činnost, datum a čas začátku, stroj, substance a očko, nebo roub. Následně se pro každou část řádku vytvoří textové pole pro zadání počtu stromků. Pro zpracování dat se používá tentýž skript. V něm se testuje, zda jsou vyplněna data formuláře a z důvodu bezpečnosti i CRFS token. Pokud jsou data v pořádku, tak se uloží do databáze.

O výpis tabulky s daty jednotlivých činností na vybraných částech se stará skript mající vybrané části řádků a filtry jako jeho parametry.

7.3.3 Statistiky

Pro zobrazení statistik bylo nezbytné vytvořit u entit repositáře s funkcemi obsahující složitější dotazy na databázi. Pokud by byly tyto dotazy řešeny v controlleru, tak by se porušily principy MVC návrhového vzoru.

7.3.4 Nastavení

U části nastavení bylo nutné vytvořit proměnnou dostupnou v celém systému. Tato proměnná je asociativní pole s klíči a hodnotami nastavenými podle tabulky „settings“ z databáze.
Aby bylo možné načíst data z databáze do globální proměnné, byla vytvořena tzv. „TWIG extension“ obsahující funkci „getGlobals“, která vrací pole s globálními proměnnými.

8 Popis systému

8.1 Uživatelé

Pro přístup do informačního systému je pro každého uživatele nutné se přihlásit. Kontrola přihlašovacích údajů se provádí s údaji v databázi. Každý uživatel má určenou roli, zajišťující přístup do různých částí systému. Nastavování uživatelů a úpravy rolí může provádět pouze uživatel s rolí administrátora.

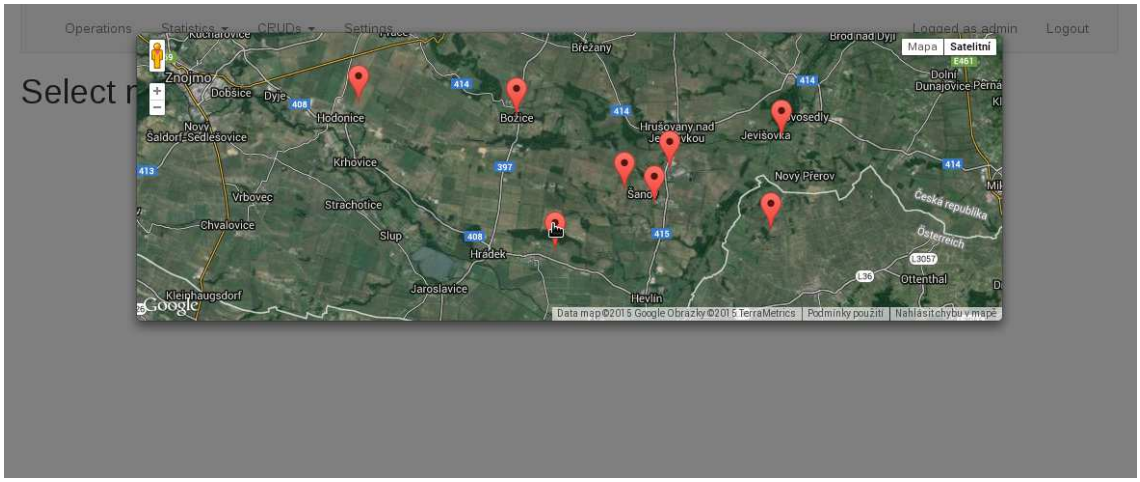
Přihlášený uživatel se eviduje také při vytvoření, nebo editaci operací, pro jeho určení při zadání chybných údajů.

8.2 Operace

Operace jsou jednou z nejdůležitějších součástí systému, po založení školky se s nimi bude pracovat nejčastěji.

V této části se provádí operace se všemi evidovanými činnostmi na školce. Mezi ně patří vytváření, editace a výpis.

Při vstupu do operací je zobrazena pouze rozbalovací nabídka s výběrem školky, na které se budou zadávat činnosti a tlačítko pro výběr požadované školky na mapě.



Obr. 4 Výběr školky pomocí mapy

8.2.1 Grafické zobrazení

Po výběru školky je na levé straně grafické zobrazení školky. Na něm jsou vidět jednotlivé řádky a jejich části oddělené kolíky. Toto zobrazení funguje tak, že se určí z databáze nejdelší řádek školky, ten je vypsán přes celou šířku části a ostatní řádky se vypočítají poměrem k tomuto řádku. V případě, že žádný řádek nemá zadaný počet stromků, jsou všechny řádky šedé a zobrazené přes celou šířku. Grafic-

ká část slouží k přibližnému zobrazení délek řádků, ale hlavně pro výběr jednoho a více řádků, nebo částí řádku, na kterých se zaeviduje určitá činnost.

8.2.2 Vytváření operací

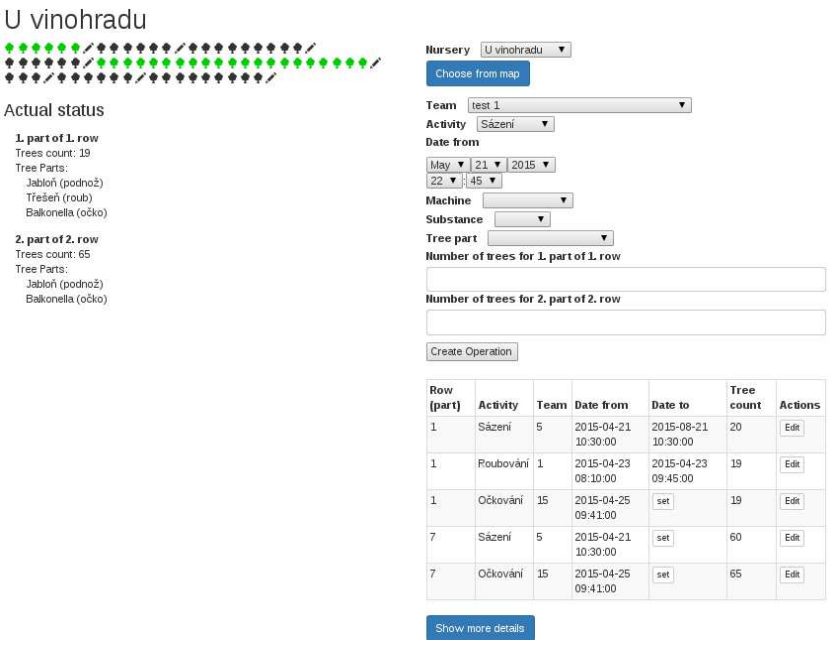
Vytvoření činnosti je možné provést v pravé části obrazovky, kde je formulář obsahující všechny potřebné údaje. Povinnými údaji jsou jen tým, činnost a datum začátku, protože u některých operací se jiné údaje neevidují. Např. při očkování je nutné k údajům ještě přidat, jaké očko bylo naočkováno a počet očkovaných podnoží na jednotlivých řádcích.

U přidávání operací na více řádků se, kromě počtů stromků na řádku, nastavují stejné údaje pro všechny řádky. Počet stromků je nutné zadávat pro každý řádek zvlášť.

Operace je možné přidávat i pro celou školku. Tato funkce slouží hlavně k evidování činností, u kterých není důležitý řádek a počet stromků na něm. Dalším případem je evidování prací, týkajících se celé školky a řádek u nich není ani možné určit, jako např. práce s postřiky.

8.2.3 Zobrazení operací

Informace o všech provedených operacích na vybraných řádcích jsou zobrazeny pod formulářem pro jejich vytváření. Tato tabulka je spíše informativní. Slouží hlavně pro nastavení data a času ukončení činnosti a editaci informací o každé činnosti.



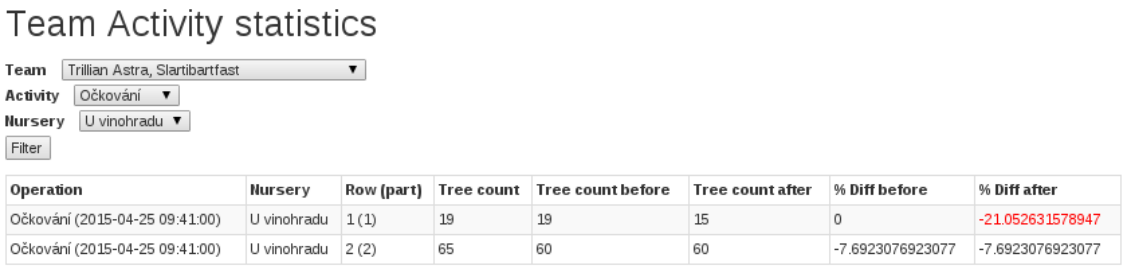
Obr. 5 Ukázka aplikace – operace

Více informací je možné zobrazit tlačítkem pod tabulkou, které otevře v novém okně informace o vybraných částek řádků pro každý řádek zvlášť, zde je také možné nastavit datum ukončení a editace.

8.2.4 Zobrazení aktuálního stavu

Pod grafickým znázorněním je zobrazen aktuální stav vybraných částí řádků. Ten obsahuje jednotlivé části stromku, ze kterých se stromky skládají a aktuální počet stromků.

8.3 Statistiky



Obr. 6 Ukázka aplikace - statistiky

Pro vytvoření uceleného pohledu na školky, nebo zaměstnance slouží část se statistikami. Všechny stránky se statistikami obsahují v horní části filtr pro určení zobrazovaných statistik. Pod ním následuje tabulka obsahující jednotlivé statistiky s červeně zvýrazněnými hodnotami přesahujícími práh, který se nastavuje v části "nastavení". Na konci tabulek je vždy souhrnná statistika pro všechny hodnoty podle filtru. V případě statistik školek je zde navíc zobrazen graf se změnami počtů stromků.

8.3.1 Statistiky školky

Ve statistikách školky lze pozorovat rozdíly vysazených (naočkovaných, naroubovaných) počtů stromků u jednotlivých odrůd. Je zde také statistika vysazených stromků a jejich aktuálního počtu.

Statistika průběžných změn počtu stromků jednotlivých odrůd je také zobrazena v přehledném grafu nad tabulkou.

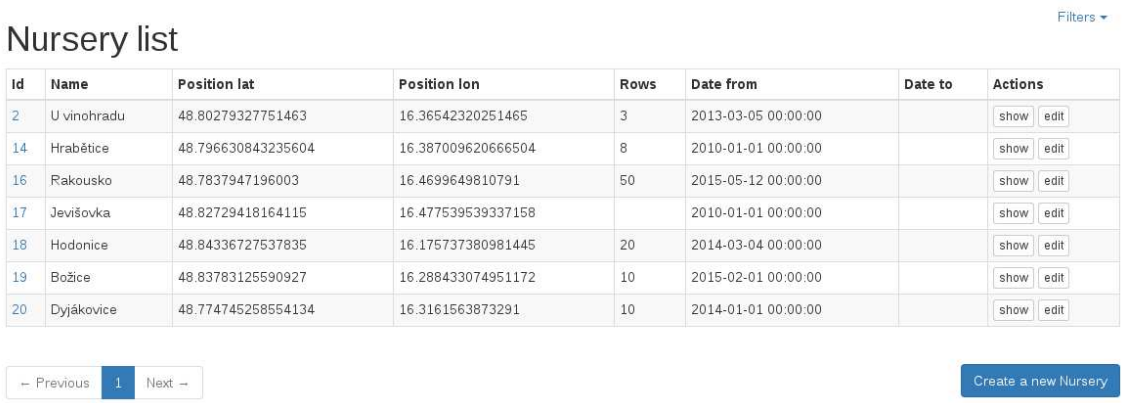
8.3.2 Statistiky týmů a zaměstnanců

Díky statistikám týmů a zaměstnanců bude ušetřeno mnoho hodin ručního počítání rozdílů stromků na jednotlivých řádcích. Hlavním účelem statistik týmů je určit, zda se neujmulo velké množství roubů, nebo oček. Pokud je těchto roubů, nebo oček mnoho, tak to bývá s největší pravděpodobností chyba jednoho z týmu, ať už vazače, nebo štěpovače. Po přezkoumání neujmutých částí a dokázání viny jednoho

z týmu se uplatňuje snížení mzdy o určenou část pro celý tým. Příčinou takové ztráty mohou být také špatná očka nebo rouby již od dodavatele, to je možné ověřit v části se statistikami školky.

8.4 CRUDy

Tato část slouží hlavně k editaci a vytváření jednotlivých položek, které se následně vybírají u vytváření jednotlivých operací.



Obr. 7 Ukázka aplikace - CRUDy

8.4.1 Zobrazení seznamu

Při vybrání podkategorie se zobrazí seznam položek. Ten je u každé podkategorie řešen jednoduchou tabulkou s vypsanými vybranými daty a na každém řádku s možností zobrazení detailu se všemi daty položky, nebo jejich editaci.

Všechny tabulky mají také filtr a stránkování pro jednodušší orientaci ve větším množství dat.

Ve spodní části je tlačítko pro vytvoření položky.

8.4.2 Vytváření a editace

Vytváření položek je řešeno za pomoci jednoduchého formuláře pro každou položku. Součástí formuláře je i ošetření povinných částí.

Editace je řešena velmi podobně, jako vytváření, jsou v ní ovšem již předvyplněna data z databáze.

8.4.3 Zobrazení položek

K zobrazení všech detailů k položce je možné se dostat z výpisu všech položek. Zde je možné položku editovat, smazat, nebo se vrátit zpět na seznam položek.

8.4.4 Školky

Nejdůležitější součástí „CRUDů“ je část s vytvářením školek. U vytváření školky je vhodné vyplnit plánovaný počet řádků pro správné zobrazení v části „operace“. Pokud se jedná teprve o plán a není počet řádků určen, tak je možné tento počet zadat později formou editace školky.

Změnou oproti ostatním částem je použití map. Při vytváření školky lze jednoduše vybrat umístění školky z mapy, která se nachází v horní části stránky. Tato mapa je použita i v editaci, kde se zobrazuje vybraná poloha a je možné ji také pouhým kliknutím změnit. Souřadnice jsou ukládány pro jednodušší výběr školky (v části operace) za pomoci mapy, kde jsou zobrazeny všechny školky.

8.4.5 Dodavatelé

Systém obsahuje také evidenci dodavatelů, to je nutné hlavně pro odlišení stejných odrůd roubů nebo oček.

8.4.6 Uživatelé a týmy

Jelikož se provádí mnoho činností na školce v týmu, a ne jednotlivě, je nutné tyto pracovní týmy a jejich činnosti evidovat. V části s CRUDy je proto vytvořena správa těchto týmů. Týmy se zde dají jednoduše vytvářet, i editovat pracovníky v týmu.

8.4.7 Nastavení

V nastavení je možné měnit různé atributy systému. Tyto údaje jsou dostupné ve všech částech systému, a nastavují se podle jejich hodnot požadované parametry.

Jsou zde zatím pouze 2 položky, při rozrůstání systému jich bude samozřejmě potřeba mnohem více. Nyní je možné nastavit výchozí pozici mapy při vytváření školek a práh, od kterého se zobrazují ve statistikách hodnoty červeně.

9 Závěr

Cílem bylo navrhnout a implementovat informační systém pro správu stromových školek. Systém zjednodušil evidenci činností prováděných na stromových školkách. Důležitým výsledkem je ukládání dat na jedno místo (do IS) bez nutnosti zbytečného přepisování dat z papíru do tabulkového procesoru.

Jelikož byla řešena část systému před vyoráváním stromků, bylo by nutné vyexportovat počty stromků do jiného systému starajícího se o třídění a prodej stromků.

10 Literatura

A Short History of JavaScript [online]. 2012-06-27 [cit. 2015-04-28]. Dostupné z: https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

ALEXIS GOLDSTEIN, LOUIS LAZARIS. HTML5 & CSS3 for the real world. 1st ed. Collingwood, VIC, Australia: SitePoint, 2011. ISBN 9780980846904.

CEDERHOLM, DAN. *CSS3 for web designers*. New York: A book apart, 2010, 125 s. Book apart, no. 2. ISBN 978-098-4442-522

GRUDL, DAVID. *Nette Framework: zvyšte svoji produktivitu*. Zdroják, [online] 2009-03-10 – 2009-06-30. [cit. 2014-05-17]. Dostupné z: http://www.zdrojak.cz/clanky/nette-framework-zvyste-svoji-produktivitu

HOGAN, B P. *HTML5 a CSS3 : výukový kurz webového vývojáře.* 1. vyd. Brno: Computer Press, 2011. 272 s. ISBN 978-80-251-3576-1.

KEITH, JEREMY. *HTML5 for web designers.* New York: A book apart, 2010, 87 s. Book apart, no. 1. ISBN 978-098-4442-508.

KOSEK, JIŘÍ. *PHP – tvorba interaktivních internetových aplikací: podrobný průvodce*. 1. vyd. Praha: Grada, 1999. 490 s. ISBN 80-716-9373-1

LAVIN, P. *PHP - objektově orientované: koncepty, techniky a kód.* 1. vyd. Praha: Grada, 2009. 211 s. ISBN 978-80-247-2137-8.

LECKY-THOMPSON, E. -- NOWICKI, S D. *PHP 6 : programujeme profesionálně.* 1. vyd. Brno: Computer Press, 2010. 718 s. ISBN 978-80-251-3127-5.

Nette Framework [online]. © 2008 - 2014 [cit. 2014-05-17]. Dostupné z: http://doc.nette.org/cs/2.1

PHP: Books about PHP [online]. © 2001 - 2014 [cit. 2014-05-17]. Dostupné z: http://www.php.net/manual/en/history.php.books.php

Symphony book [online]. 2014 [cit. 2014-05-18]. Dostupné z: http://symfony.com/pdf/Symfony_book_2.4.pdf?v=4

Usage of server-side programming languages for websites [online]. 2009-2015 [cit. 2015-04-28]. Dostupné z: http://w3techs.com/technologies/overview/programming_language/all

TÖGL, JAN. *Nette Framework*. Praha 2012. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií. Vedoucí práce Ing. Jarmila Pavlíčková

11Seznam obrázků

Přílohy

A Zdrojový kód

Zdrojový kód se nachází na přiloženém CD.