Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Informační systém pro IT společnost

2015

Ondřej Směták

Vedoucí práce: RNDr. Arnošt Večerka

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Ondřej Směták
Název práce:	Informační systém pro IT společnost
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2015
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	RNDr. Arnošt Večerka
Počet stran:	40
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Ondřej Směták
Title:	Information system for IT company
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2015
Study field:	Applied Computer Science, full-time form
Supervisor:	RNDr. Arnošt Večerka
Page count:	40
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Efektivní řízení obchodní společnosti za pomoci na míru vytvořeného informačního systému. Takového cíle si klade dosáhnout vyvinutá aplikace určená pro nasazení v oblasti IT. Nabízí nástroje k produktivní komunikaci se zákazníkem, správu odvedené práce, fakturační systém i evidenci jednotlivých požadavků klientů. Na základě dostupných dat aplikace vytváří statistické přehledy, které vedení společnosti využívá k rychlému seznámení s aktuálním stavem ve firmě nebo na jejich základě provádí strategické plánování dalšího období. Informační systém se zaměřuje na konzistentní uživatelské rozhraní, jehož používání se jeví díky dostatku kontextových nápověd dostatečně srozumitelné, a to z pohledu všech skupin uživatelů.

Synopsis

Efficient company management thanks to bespoke designed information system. That objective aims to achieve developed application designed for deployment in IT industry. Information system provides tools for productive communication with customers, managing finished work, integrated billing system and keeps track of individual client's requirements. Based on available data, the application generates statistical reports that company management uses to quickly acquaint the current state of the company or makes decisions regarding strategic planning for the next time period. Information system also focuses on consistent user interface that benefits from contextual tips and will be assessed as easy to understand from the perspective of all user groups.

Klíčová slova: informační systém, Nette, PHP, objektově relační mapování

Keywords: information system, Nette, PHP, object-relational mapping

Děkuji RNDr. Arnoštu Večerkovi za ochotu a odborné vedení poskytnuté v průběhu zpracování bakalářské práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úloha informačního systému

Podnikový informační systém zastává na poli vnitřních firemních procesů svoji nezaměnitelnou roli. Dokáže zefektivnit jednotlivé firemní procesy a přinést vedení společnosti odpovědi na palčivé otázky, a to v reálném čase. V případě vyvinutého informačního systému pro IT společnost[1] si však můžeme dovolit jít ještě dále a zapojit do vnitrofiremních procesů další neméně důležité subjekty každého obchodního styku – samotné zákazníky. Dle teorie marketingu [1] tedy budeme zasahovat do:

- vnitřního prostředí (lidské zdroje a organizace řízení)
- vnějšího prostředí (zákazníci)

S ohledem na uvedené informace nic nebrání tomu, abychom rovněž označovali vyvinutý informační systém jako nástroj k *řízení vztahů se zákazníky* (běžněji zkracováno jako CRM z anglického „customer relationship management“).

1.1 Výhody oborového řešení

Pokud má informační systém zodpovědně přispívat k zefektivnění provozu firmy, musí umět pružně reagovat na specifické potřeby dané oblasti podnikání, ve které je nasazen [1]. Právě proto vytvořený IS směrovaný do oblasti IT[2] přináší vybrané funkce, které by patrně v jiných oborech lidské činnosti nenašly své uplatnění. My však cílíme na malý až střední podnik, který se typicky zabývá dodávkou desktopových aplikací, serverových řešení, webdesignu, webových portálů nebo e-shopů. Mezi nejčastější zaměstnance zmíněné firmy bude patřit např. programátor, tester, grafik nebo osoba zajišťující uživatelskou podporu. Vedení popisované modelové společnosti ocení efektivní organizaci zadané a již odvedené práce. Všem zmíněným subjektům se musí IS přizpůsobit a stejným flexibilním způsobem se chovat i k zákazníkům. Ujasněme si proto na následujících řádcích potřeby a očekávání uživatelů podle jejich jednotlivých rolí.

1.1.1 Zaměstnanec

Z pohledu zaměstnance získává vytvořený IS na významu v momentě, kdy pracovník řeší, jakému úkolu se má aktuálně věnovat. K dílčímu požadavku na zpracování jsou kromě jeho zadání vedeny i další informace. Jestliže zaměstnanec potřebuje upřesnit zadání, může jednoduše kontaktovat osobu nejpovolanější – samotného zákazníka.

[1] Dále v textu zkracováno na IS z anglického „information system“ a potažmo i českého označení „informační systém“.

[2] Zkratka „IT“ je běžně používána pro označení odvětví zabývajícího se informačními technologiemi.

1.1.2 Zákazník

Klient od IS očekává seznámení s aktuálním stavem prací. Zajímá jej, na jakém jeho požadavku se momentálně pracuje a kdy bude přibližně dokončen. V úvahu je třeba vzít i ekonomickou stránku věci. Zákazník se přirozeně stará o obchodní hledisko jeho projektu, a proto ocení vyčíslení přibližného rozsahu příští faktury[3], díky čemuž může snadno kontrolovat využití jím vyčleněného měsíčního rozpočtu[4].

1.1.3 Vedení společnosti

Možnost přístupu k aktuálním podkladům pro strategické rozhodování kladně ohodnotí vrcholový management většiny firem [1]. Klíčová je dostupnost informací o aktuálním stavu dosud nedokončené práce a rozsahu vedených požadavků stejně jako odhadů budoucích faktur a mezd, tedy potažmo předpokládaného zisku členěného podle jednotlivých klientů a jejich projektů.

1.1.4 Obecné požadavky

Seznámili jsme se s potřebami jednotlivých typů uživatelů. Praktické nasazení IS dále ukázalo na potřebu nezanedbat vytvoření jednoduše použitelného uživatelského rozhraní. Klient od IS očekává maximálně snadné provedení veškerých úkonů, jinak ztrácí motivaci k jeho používání. Rozhodně není vhodné přijmout chybnou domněnku, že mezi zákazníky IT společnosti budou častokrát patřit osoby s alespoň povrchními znalostmi z oblasti informačních technologií. V praxi se ukázalo, že tomu bývá přesně naopak, a proto byl během vývoje IS kladen důraz na přehlednou navigaci a poskytnutí kontextové nápovědy k jednotlivým funkcím.

1.2 Výběr platformy

Po vytvořeném IS požadujeme, aby k němu byl přístup co nejkomfortnější. Z dostupných možných řešení byla zvolena varianta webové aplikace. K jejímu spuštění ze strany uživatele vyžadujeme pouze webový prohlížeč, který je standardní součástí nejrozšířenějších operačních systémů. Vybraná platforma s sebou sice nese nutnost připojení k internetu, s tímto omezením bychom se však potýkali i v případě jiných platforem, protože již v části 1.1 jsme narazili na nutnost poskytovat vždy aktuální a přesné informace, což bez komunikace s centrálním úložištěm dat lze jen obtížně zajistit. Výhody webové aplikace lze spatřit v dostupnosti prostřednictvím mobilních telefonů a tabletů. I když tato zařízení typicky nebudou využívána během zadávání dat, mohou se na ně spolehnout

[3] V obchodním styku lze termínem faktura [2] označit účet za odvedenou práci, dodané zboží nebo provedené služby. Faktura kromě dalších náležitostí obsahuje soupis jednotlivých položek, celkovou částku k úhradě a datum splatnosti. Pokud v textu není uvedeno jinak, pojmem faktura vždy rozumíme fakturu vystavenou společností používající vyvinutý IS.

[4] V praxi je běžně používán anglický termín „budget“.

uživatelé ve chvíli, kdy se potřebují seznámit s aktuálním stavem (tedy budou informace pouze číst). Ve velmi snadný úkon se dále proměňuje případná distribuce aktualizované verze aplikace.

2 Technické řešení

Dle sekce 1.2 je vytvořený IS situován do podoby webové aplikace. K implementaci byl použit skriptovací programovací jazyk PHP. Ten ke své interpteraci vyžaduje k tomu určený webový server. V prostředí linuxové distribuce lze například použít server Apache.

2.1 Nette Framework

Jako hlavní opěrný bod pro vývoj webové aplikace byl zvolen Nette Framework pro PHP. V současné době se o rozvoj Nette Frameworku stará skupina jednotlivců sdružená v rámci Nette Foundation [3]. Původním tvůrcem Nette Frameworku je český programátor David Grundl.

Mezi hlavní přednosti Nette Frameworku patří [4] :

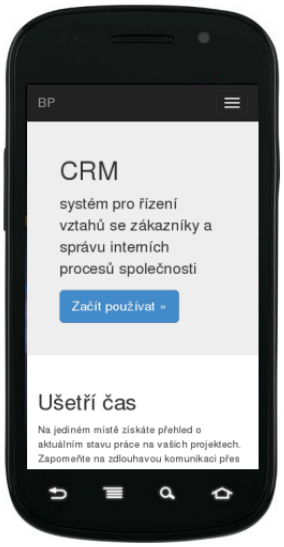
- zaměření na bezpečnost a eliminaci potencionálních bezpečnostních chyb
- podpora objektově orientovaného programování v rámci PHP 5
- propracované ladící nástroje
- použitá licence umožňuje komerční nasazení
- celková koncepce frameworku pamatuje na možnost budoucího rozšíření a dokáže rozčlenit práci v rámci týmu, ale přitom se drží v duchu praktiky KISS[5] i MVC [6]

[5]KISS pochází z anglického „keep it short and simple“. Jde o návrhový vzor aplikující ideu, že většinu systémů lze snadněji udržovat v tom případě, když jsou vnitřně koncipovány jednoduše.

[6]Zkratka MVC vznikla z anglického označení „model-view-controller“. Jedná se o architekturu pro implementaci uživatelského rozhraní, která od sebe vzájemně odděluje přístup k datům (model), jejich prezentaci uživateli (view) a interakci na provedené události (controller). V terminologii Nette Frameworku je controller označován jako presenter, a proto se lze někdy také setkat se zkratkou MVP, ta se však principiálně od MVC nijak významně neliší.

2.2 Bootstrap framework

Frontend[7] framework Bootstrap symbolizuje kolekci připravených komponent pro použití v rámci HTML[8] a CSS[9]. Kromě jednotlivých částí pro nasazení v prostředí webové stránky je k dispozici i startovací minimalistická šablona. Ta používá techniku označovanou jako CSS reset[10], která přispívá ke konzistentnímu zobrazení finální stránky napříč webovými prohlížeči. Mezi další nezanedbatelnou výhodu frameworku Bootstrap patří plná podpora pro vytvoření responzivního webového designu, který se umí přizpůsobit zařízení, na kterém je zobrazován. V případě menšího rozlišení (typicky např. na mobilním telefonu) automaticky dojde k přeskládání prvků minimalistickým způsobem, aby používání webu bylo i na omezené ploše co nejjednodušší. V případě vyššího rozlišení (např. na obrazovce běžného počítače) si naopak webová stránka může dovolit plně využít nabídnutý prostor.



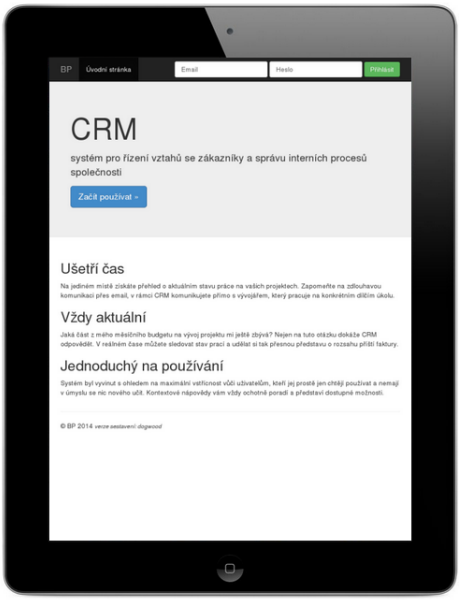
Obrázek 1: Úvodní stránka informačního systému zobrazená na mobilním zařízení.

[7]Pojmem „frontend“ se v prostředí webu označuje ta část aplikace, která je viditelná návštěvníkům. Typicky se tedy jedná o samotné uživatelské rozhraní.

[8]Zkratka „HTML“ vznikla z anglického označení „HyperText Markup Language“. Jedná se o značkovací jazyk určený k tvorbě hypertextových dokumentů, které používá systém World Wide Web (WWW).

[9]Termín „CSS“ označuje jazyk „Cascading Style Sheets“ určený k popisu vzhledu jednotlivých elementů na webových stránkách vytvořených v jazyce HTML.

[10]CSS reset slouží ke konzistentnímu zobrazení prvků webové stránky napříč všemi nejpoužívanějšími webovými prohlížeči. Cílem je dosáhnout toho, aby stránka vypadala ve všech prohlížečích prakticky shodně. Potřeba techniky CSS reset vyvstala v momentě, kdy rozdílné webové prohlížeče začaly vykreslovat určité prvky svým vlastním osobitým způsobem.



Obrázek 2: Shodná úvodní stránka zobrazená na tabletu.

2.3 Lean Mapper

Lean Mapper představuje nástroj pro objektově relační mapování[11], který využívá knihovnu dibi[12]. Značnou výhodou a důvodem začlenění popisované knihovny do IS je schopnost generovat SQL dotazy[13] takovým způsobem, že dochází k získání množiny výsledků najednou, tedy zpravidla s pomocí jednoho dotazu, namísto pokládání individuálního dotazu pro každý objekt. Uvedená skutečnost platí i v případě, kdy objekt v OOP[14] vyžaduje v databázi rozprostření do více relací (tabulek).

2.3.1 Motivační příklad

Výhodu knihovny Lean Mapper si ukážeme na konkrétním příkladě. V rámci IS máme definovánu entitu[15] klienta a projektu. Mezi těmito entitami platí vztah 1:N, tedy jinými slovy jeden klient může mít v IS vedených více projektů. Podívejme se nyní na konkrétní definici obou entit v jazyce PHP.

[11] Častěji označováno zkratkou ORM. Jde o techniku sloužící k automatickému převodu objektu mezi zvoleným objektově orientovaným jazykem a databází založenou na relačním modelu. Nutno podotknout, že zmíněnou konverzi je možno provést oběma směry.

[12] Knihovna dibi v rámci PHP 5 zapouzdřuje přístup k samotné databázi a přináší metody schopné skládat výsledné SQL dotazy. Není bez zajímavosti, že tento nástroj patří pod křídla Nette Foundation, tedy stejné organizace, která stojí i za Frameworkem Nette.

[13] Structured Query Language zkracovaný na „SQL“ je dotazovací jazyk, který slouží pro práci se záznamy v relačních databázích.

[14] Zkratka pro „objektově orientované programování“.

[15] V objektově relačním mapování slouží entita k popisu objektu z reálného světa. V relační databázi povětšinou entita nabývá podoby řádku v konkrétní tabulce, v aplikaci se naopak obvykle jedná o instanci zvolené třídy.

```php
<?php
/**
 * @property Project[] $projects m:belongsToMany
 */
class Client extends BaseEntity
{

}
?>
```

Zdrojový kód 1: Definice entity Client v souboru ClientEntity.php

```php
<?php
/**
 * @property Client $client m:hasOne
 */
class Project extends BaseEntity
{

}
?>
```

Zdrojový kód 2: Definice entity Project v souboru ProjectEntity.php

Z uvedených zdrojových kódů 1 a 2 lze vidět, že knihovna klade na programátora minimální požadavky. Stačí vytvořit novou třídu dědící ze základního předka (v tomto případě pojmenovaném jako **BaseEntity**), který dodává případné potřebné metody pro všechny potomky. V komentáři pomocí anotace **@property** definujeme v tomto pořadí návratový typ, název interní proměnné a typ relace. Třída Client tedy vnitřně pracuje s proměnou **$projects**, která obsahuje pole entit datového typu **Project**. Díky dodržování konvencí v oblasti pojmenovávání databázových tabulek a jejich sloupců Lean Mapper po uvedení příznaku relace (např. **m:belongsToMany** pro typ 1:N) automaticky dokáže vygenerovat příslušný SQL dotaz. Ve zdrojovém kódu IS se proto zapsané SQL dotazy vyskytují minimálně. Pokud bylo přistoupeno k ručnímu zápisu dotazu namísto jeho generování, bylo to z důvodu zajištění maximální efektvitiy v nejpoužívanějších částech systému, kdy je zbytečné opětovně generovat dotaz, jehož struktura se v průběhu používání IS vůbec nemění.

Lean Mapper s uvedenými entitami umožňuje pracovat za pomoci getterů[16] a setterů[17], jejich existence je možná díky tzv. magickým metodám **___get** a **___set**. Ty zpřehledňují zápis zdrojového kódu, protože umožňují pracovat se

[16]Getter je metoda vracející hodnotu neveřejné proměnné zapouzdřené v rámci instance určité třídy.

[17]Setter je metoda sloužící k nastavení neveřejné hodnoty zapouzdřené v instanci konkrétní třídy.

soukromými (**private**) a chráněnými (**protected**) proměnnými vybrané třídy takovým způsobem, kdy se navenek jeví, že se jedná o veřejně dostupné proměnné (**public**).

```
<?php
$client->name = "Jan"; //ekvivaletní s $client->setName("Jan");
echo $client->name; //ekvivaletní s $client->getName();
?>
```

Zdrojový kód 3: Magické metody v PHP 5

Z výše popsaného vyplývá, že Lean Mapper vybrané operace na pozadí převádí do požadovaného kódu v PHP 5, čímž zamezuje opakování velmi podobného kódu (např. v případě již jmenovaných getterů a setterů, které se v podstatě běžně liší jen v členské proměnné, ke které se přistupuje). Programátor však zmíněným chováním není nijak omezen, protože kdykoliv v případě nutnosti může generované metody v duchu dědičnosti překrýt svým vlastním kódem. Pro lepší představu se podívejme na úplnou konkrétní implementaci jedné z entit.

```
<?php
namespace BP\Model;

/**
 * @property int $id
 * @property boolean $vat
 * @property string $name
 * @property string $street
 * @property string $city
 * @property string $zip_code
 * @property string $country
 * @property string $identification
 * @property boolean $identification_validate
 * @property string|NULL $identification_vat
 * @property string $email
 * @property string $phone
 * @property User[] $users m:belongsToMany
 * @property Project[] $projects m:belongsToMany
 */
class Client extends BaseEntity
{

}
?>
```

Zdrojový kód 4: Kompletní implementace třídy Client nacházející se v souboru ClientEntity.php

Vidíme, že pro práci se základními gettery a settery nám skutečně pouze stačí v rámci anotace **@property** definovat název proměnné a její datový typ. V případě relačních odkazů na jiné entity je situace v podstatě stejná, pouze uvedeme navíc příznak relace.

V době vzniku tohoto textu (listopad 2014) Lean Mapper disponoval nevýhodou v podobě chybějící podrobné dokumentace. I když úvodní stránka projektu jasně uvádí, že „Lean Mapper není nezdokumentované cosi“ [9], pravdou je spíše opak. Webová stránka projektu sice obsahuje popis základních technik, informace o pokročilejších funkcích je však mnohdy nutno hledat přímo ve zdrojovém kódu knihovny. Drobné potíže dále nastaly během začlenění knihovny do Nette Frameworku, který pro práci s databází používá vlastní vrstvu **Nette\Database**. Bylo nutno upravit konfiguraci frameworku takovým způsobem, aby pro práci s databází využíval právě knihovnu Lean Mapper.

2.4 Další technologie

Během vývoje IS byly použity i další technologie. Následuje jejich výčet.

2.4.1 Javascriptová knihovna jQuery

Nutnou prerekvizitu pro použití frameworku Bootstrap (2.2) symbolizuje knihovna jQuery [11] zapouzdřující technologii JavaScript pro interakci se stránkou na straně webového prohlížeče.

2.4.2 mPDF

Knihovna mPDF [12] dokáže vygenerovat soubor ve formátu PDF[18] z HTML. Proto našla své uplatnění v rámci IS, kde slouží k exportu faktury.

2.4.3 Eciovni

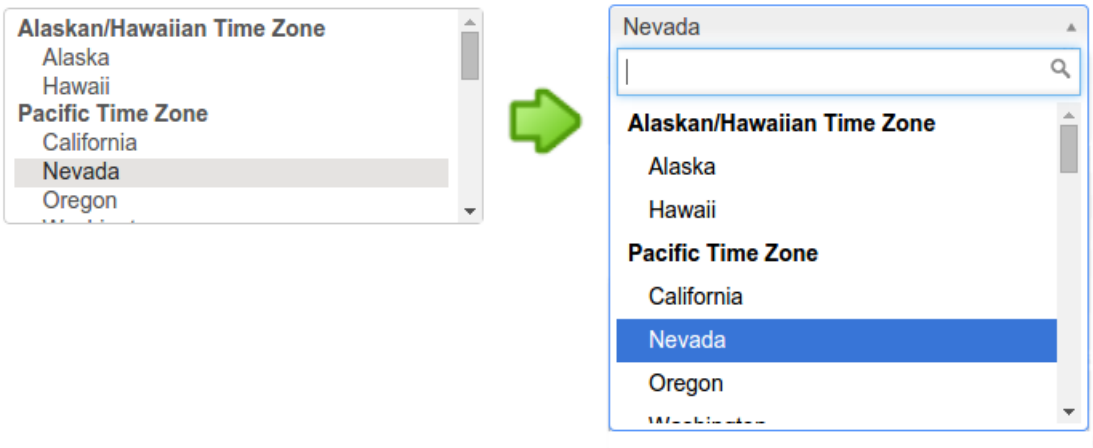
Doplněk Eciovni [10] pro Nette Framework (2.1) využívá knihovnu mPDF (2.4.2) a usnadňuje samotné generování faktur. Autorem doplňku je český programátor Ondřej Brejla.

2.4.4 Select2

Projekt Select2 [13] dokáže v prostředí webové aplikace transformovat běžný rozbalovací seznam ve vstupní prvek s uživatelsky příjemným chováním. To se projevuje v podobě interaktivního napovídání během zadávání z množiny přípustných hodnot. Zmíněná proměna v nový vstupní prvek probíhá na úrovni webového prohlížeče a k jejímu provedení je vyžadována Javascriptová knihovna jQuery (2.4.1).

[18] Zkratka pro „Portable Document Format“. Jde o souborový formát vytvořený společností Adobe, který byl vyvinut s ohledem na maximální přenositelnost napříč různými zařízeními.

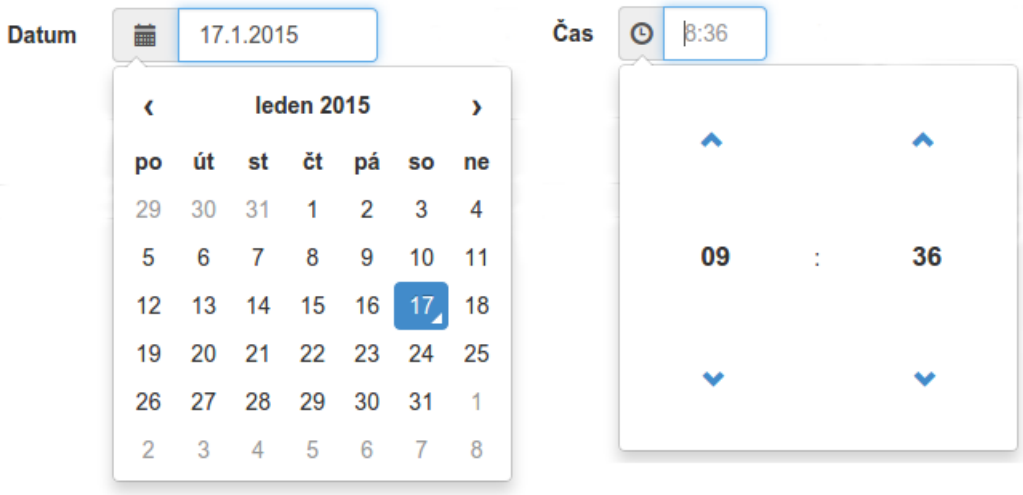
Implementaci knihovny Select2 bylo v případě IS nutno obohatit o specifickou definici kaskádových stylů pro framework Bootstrap (2.2), aby bylo dosaženo konzistentního uživatelského rozhranní.



Obrázek 3: Rozbalovací seznam upravený knihovnou Select2

2.4.5 DateTimePicker

Knihovna DateTimePicker [14] určená pro framework Bootstrap (2.2) přináší nástroj pro uživatelsky pohodlné zadávání data a času. Během editace předem vymezených polí dochází k zobrazení kontextového pomocníka v podobě kalendáře a číselníku.



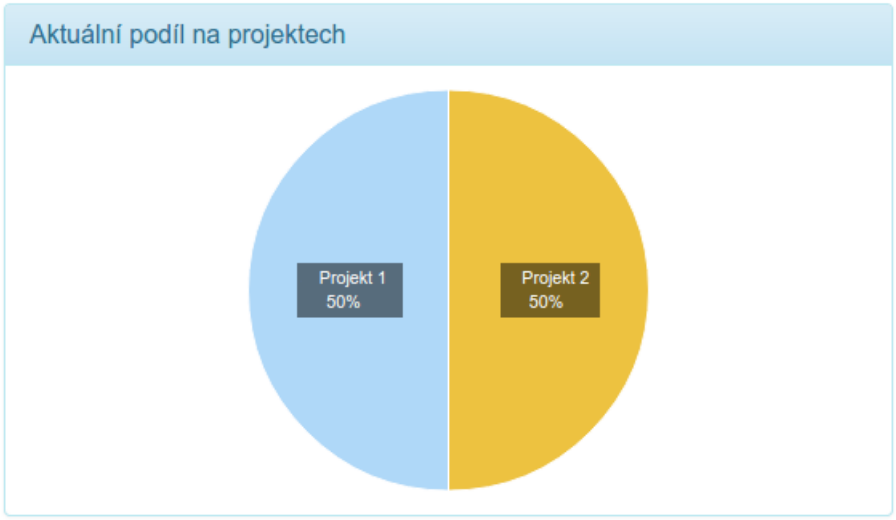
Obrázek 4: Výběr data a času pomocí knihovny DateTimePicker

2.4.6 Flot

Vyvinutý IS zobrazuje v rámci sekce se statistikami různorodé informace, které lze mnohdy prezentovat i v podobě grafu. Jeho vygenerování můžeme principiálně řešit dvěma způsoby:

- vygenerování grafu na straně webového serveru a následné odeslání klientovi v podobě běžného obrázku
- vykreslení grafu z dostupných údajů přímo na straně klienta

Z dostupných prezentovaných možností byla zvolena druhá varianta, tedy vygenerování na klientské straně, a to s pomocí JavaScriptu. Mezi hlavní důvody patřila snaha o omezení zbytečného zatěžování serveru a možnost nabídnout uživateli interaktivní grafové zobrazení. Jako nástroj k dosažení vytčeného cíle byla zvolena knihovna Flot [15], která pro svoji práci vyžaduje jQuery (2.4.1).



Obrázek 5: Graf vykreslený pomocí knihovny Flot

3 Návrh systému

3.1 Struktura projektu

Samotná interní struktura aplikace a její logika je vymezena použitím Nette Frameworku, jenž nastiňuje základní mantinely, kterých je vhodné se držet. Není pochopitelně nutné je plně dodržovat, protože však klademe důraz na znovupoužitelnost a budoucí možný rozvoj projektu, budeme se zásad, s nimiž framework operuje, držet.

Podívejme se nejprve na samotnou adresářovou strukturu.

- **app** – obsahuje aplikační logiku

  - **components** – umístění jednotlivých komponent (3.4)
  - **config** – obsahem jsou konfigurační soubory aplikace
  - **forms** – složka sdružuje třídy popisující formuláře (3.11)
  - **localization** – uchovává XML soubory vyžadované pro překlad do cílového jazyka (3.3)
  - **model** – s tímto umístěním pracuje objektově relační mapování (2.3), které v architektuře MVC zastupuje komponentu datového modelu
  - **other** – ostatní soubory obsahující různé pomocné nástroje
  - **presenters** – klíčová složka MVC (MVP) v podobě jednotlivých presenterů
  - **router** – součást Nette Frameworku sloužící k dynamickému generování URL [19]
  - **templates** – veškeré náležitosti vyžadované pro prezentaci informací uživateli jsou umístěny právě v této složce, která tak v případě MVC nese zodpovědnost za uživatelské rozhranní

- **log** – zde nalezneme různorodé informace určené převážně pro ladění, které se automaticky generují v reálném čase při běhu aplikace
- **temp** – složka obsahující dočasné soubory
  - **btfj.dat** – obsahuje žurnál[20] souborového systému, jenž uchovává data uložená v cache[21]
  - **latte** – tuto složku využívá šablonovací systém Latte[22] ke generování obsahu, který bude následně na webu zobrazen
  - **Nette.Configurator** – zde se nachází aktuální konfigurace aplikace
  - **_Nette.RobotLoader** [23] – během automatického načítání tříd vystává potřeba vyrovnávací paměti pro efektivní práci tohoto nástroje
- **vendor** – na tomto místě nalezneme knihovny třetích stran

[19] Zkratka „URL“ označuje termín „Uniform Resource Locator“ sloužící k jednoznačné identifikaci umístění zdroje.

[20] Žurnál chrání data před ztrátou jejich integrity. Obsahuje záznamy o akcích prováděných na úrovni souborového systému, s jejichž pomocí lze v případě neočekávané chyby uvést souborový systém zpět do konzistentního stavu.

[21] Cache (česky „mezipaměť“) urychluje přístup k často používaným datům. Není proto nutno opětovně spouštět náročný výpočetní proces, pokud jeho výsledek již máme v cache uložen.

[22] Latte jako součást Frameworku slouží k vykreslování výstupu a dokáže pracovat se šablonami, tedy předpřipravenými zápisy HTML, do nichž je dynamicky doplňován příslušný obsah.

[23] Nette Framework prochází adresářovou strukturu aplikace a automaticky načítá všechny soubory (potažmo třídy) potřebné pro běh webu. Tato technika je označována jako „Autoloading“.

- **nette** – jak už název napovídá, nachází se zde jádro Frameworku Nette
- **www** – v této složce se nachází všechny soubory, které jsou volně přístupné na webovém serveru, což přispívá ke zvýšení bezpečnosti aplikace, protože máme jednoznačně oddělenou složku s veřejně zobrazitelným obsahem. Typicky se zde nachází nejrůznější obrázky, kaskádové styly (soubory s koncovkou .css) a JavaScriptové funkce (koncovka .js)

3.2 Šablona pro emailové zprávy

V praxi se ukázalo, že není úplně triviální zajistit konzistentní zobrazení HTML těla emailové zprávy napříč používanými poštovními klienty. Nejzásadnější komplikace během vývoje způsobovaly novější verze aplikace *Microsoft Office Outlook*, které jsou ovšem nejen v korporátním prostředí běžně využívány. Důvod je třeba hledat v omezené podpoře renderování HTML v kombinaci s CSS (viz. [6]).

Popsaný problém byl vyřešen nasazením šablony z projektu Email Blueprints [7], který pochází od společnosti MailChimp (http://mailchimp.com/) zabývající se odesíláním emailů pro marketingové účely.

Po nasazení šablon z projektu Email Blueprints se objevily potíže se zobrazením v rámci populární [8] služby GMail, která ignoruje CSS vložené v hlavičce zprávy. Finální řešení poskytl nástroj CSS Inliner Tool (opět z dílny společnosti MailChimp dostupný na http://templates.mailchimp.com/resources/inline-css/), který kaskádové styly dovedl přiřadit každému HTML tagu s pomocí tzv. inline zápisu[24]:

```
1   <td style="border-collapse: collapse;">
```

Zdrojový kód 5: Ukázka použití inline CSS

3.3 Překlad do cizího jazyka

Aplikace je plně připravena na případný překlad do zvoleného cizího jazyka. Při svém spuštění si načítá XML soubor obsahující příslušné jazykové definice. Kvůli zpřehlednění zápisu jsou související záznamy seskupeny do pojmenovaných sekcí. Text k překladu se poté skládá v tomto pořadí z názvu zmíněné sekce, lomítka a samotného textového řetězce (klíče) pro překlad.

Použití v rámci šablonovacího systému Latte je poté jednoduché – stačí využít tzv. podtržítkové makro[25].

[24] Termínem „inline zápis“ se označuje přiřazení stylu přímo konkrétnímu prvku v HTML.

[25] Podtržítkové makro slouží k překladu zadaného textu. Předaný textový řetězec je vyhledáván v seznamu dostupných jazykových definic. V případě shody je zobrazen specifikovaný překlad, v opačném případě dochází k výpisu původního vstupu.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<localization language="Czech" iso="cs">
  <section name="COMMON">
    <t from="PASSWORD" to="Heslo" />
  </section>
</localization>
```

Zdrojový kód 6: Ukázka XML souboru obsahujícího jazykové definice

```
<!- původní vstup -->
<p>{_"COMMON/PASSWORD"}</p>

<!- výsledný vygenerovaný HTML kód po použití makra -->
<p>Heslo</p>
```

Zdrojový kód 7: Použití podtržítkového makra

Další práce spojená s překladem aplikace by se dále nesla v duchu vytvoření samostatných šablon pro jednotlivé emailové zprávy (ve složce **app/templates/@emails**) a definici předmětu i dalších nezbytných náležitostí každé takové emailové zprávy.

3.3.1 Technické řešení překladu textu

Klíčovou roli v překladu textu hraje třída **XmlTranslator**. V systému dochází ke konstrukci objektu zmíněné třídy, během něhož je nutno ve formě argumentů uvést název XML souboru s jazykovými definicemi a pro snadnou identifikaci aktuálního jazyka i jeho ISO zkratku[26]. Během čtení XML je budováno asociativní pole[27] obsahující jako klíč vstup určený k překladu, zatímco hodnotu zastupuje příslušný překlad. Následně již nic nebrání provedení samotného překladu s pomocí metody **translate** (její přítomnost vynucuje rozhraní[28] **Nette\Localization\ITranslator**). Dotyčná metoda přebírá v argumentu text k překladu, z něhož zjistí pojmenování sekce. Podle něj přistoupí k příslušnému asociativnímu poli a klíč ze vstupu je následně použit k získání finálního překladu v cílovém jazyce.

[26] Pro účely IS se používá ISO 639-1. Tento standard umožnuje jazyky identifikovat podle dvoupísmeného kódu. Češtinu například určuje zkratka „cs“.

[27] Asociativní pole se od běžného pole liší tím, že k indexaci prvků nepoužívá přirozená čísla (včetně nuly), ale textové řetězce.

[28] Rozhraní (anglicky označováno jako **interface**) je programová konstrukce specifikující nutnou přítomnost určitých metod vně třídy. Rozhraní neobsahuje těla těchto metod, ale typicky udává jejich název, návratový typ a předávané argumenty.

3.4 Komponenty

V terminologii Frameworku Nette je pojmem „komponenta“ označována předpřipravená část webové stránky, kterou lze opakovaně na libovolných místech využívat. Přesněji se jedná o třídu s definovanou vlastní šablonou. Zmíněná třída typicky ve svém konstruktoru nebo metodě **render** přebírá argumenty potřebné pro korektní zobrazení. Výhodu v použití systému komponent lze spatřit ve zjednodušení zdrojového kódu. Pro vykreslení stačí vytvořit instanci kýžené komponenty, předat ji argumenty (pokud nějaké vyžaduje) a v šabloně, na místě kde vyžadujeme zobrazení komponenty, použít stručné makro.

```
1  <?php
2  class ExamplePresenter
3  {
4    public function createComponentAddress()
5    {
6      //Vytvoření komponenty pro výpis adresy
7      return new \BP\Address($this->getTranslator());
8    }
9  }
10 ?>
```

Zdrojový kód 8: Vytvoření instance komponenty

V Presenteru[29] tedy disponujeme objektem dané komponenty. Ten v šabloně zobrazíme s pomocí makra **control**. V našem případě se jedná o šablonu pro výpis adresy, musíme proto komponentě předat objekt třídy, který v sobě potřebné údaje pro sestavení adresy obsahuje (zde jako proměnná **record**).

```
1    {control address, $record}
```

Zdrojový kód 9: Použití vytvořené komponenty v šabloně

Obrázek 6: Adresa vykreslená s pomocí komponenty

[29]Presenter ve Frameworku Nette vždy získává podobu třídy. Ta slouží k řízení aplikační logiky. Zpracovává požadavky od uživatele, podle nich volá jednotlivé akce a žádá o vykreslení příslušné šablony.

3.5 Nástěnka

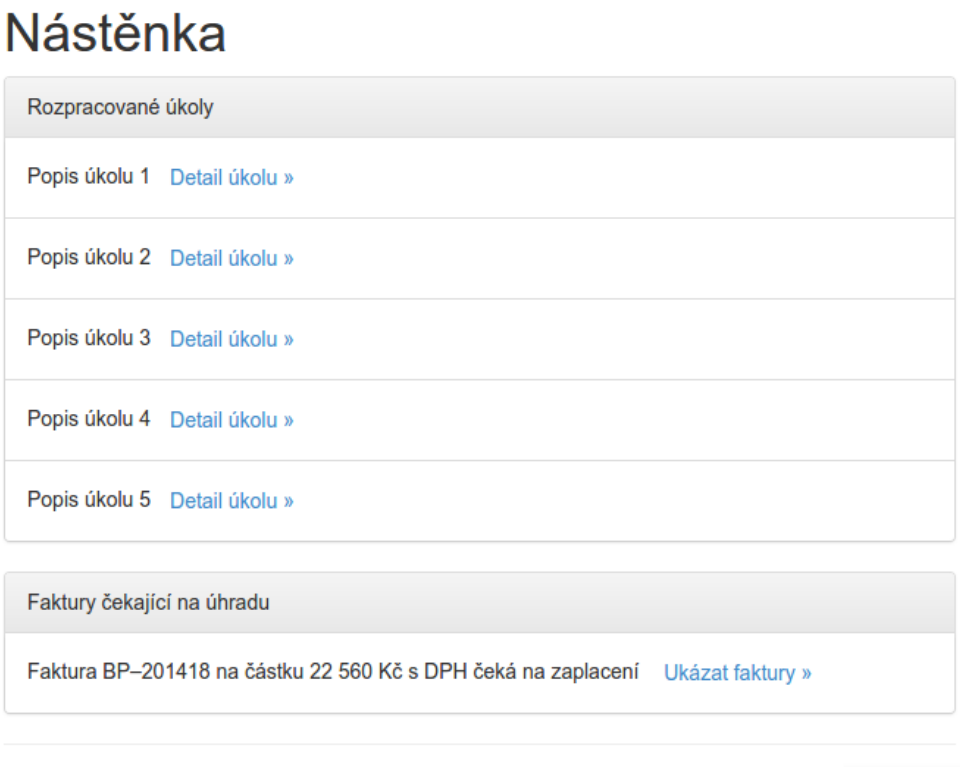
Nástěnka symbolizuje místo, které každý uživatel spatří bezprostředně po svém přihlášení a nalezne zde údaje, které jej mají v co nejkratším čase seznámit s aktuálním děním. Prezentované informace lze rozdělit do několika rozdílných kategorií. Ty v IS zastupují třídy **InvoiceReportFactory**, **PayoutReportFactory**, **StaffReportFactory** a **TaskReportFactory**. Všechny zmíněné třídy dědí ze společného předka, kterým je abstraktní třída **BaseReportFactory** obsahující abstraktní metody starající se o vygenerování informací pro zobrazení v rámci jednotlivých uživatelských rolí (metody se jmenují **forAdmin()**, **forStaff()** a **forClient()**). **BaseReportFactory** dále obsahuje abstraktní metodu **getWallTitle** vracející titulek pro skupinu zpráv, které budou na nástěnce zobrazeny. Ve třídě **WallPresenter** dochází k použití zmíněných vytvořených tříd schopných generovat jednotlivé informační hlášky. Na objekt dané třídy je zavolána metoda podle uživatelské role aktuálně přihlášeného uživatele (tj. v případě klienta půjde o **forClient()**, u zaměstnance se jedná o **forStaff()**, administrátorovi přehled vygeneruje metoda **forAdmin()**). Dostupné zprávy z rozdílných skupin jsou následně sloučeny, a to s náhodným pořadím (aby nástěnka nepůsobila konstantně a více motivovala uživatele k prostudování jednotlivých informací). V případě, kdy budoucí vývoj ukáže nutnost implementovat novou skupinu zpráv, bude její vývoj přímočarý. Postačí vytvořit novou třídu dědící z **BaseReportFactory**, dopsat těla abstraktních metod a ve třídě **WallPresenter** vytvořit novou instanci takové třídy.

3.6 Autorizace

Autorizace je proces, kdy uživateli povolujeme některé akce. V případě IS se jedná o klíčovou událost, protože v systému nám figuruje několik typů uživatelských účtů s diametrálně odlišným přístupem k informacím (například není žádoucí, aby zaměstnanec znal hodinové sazby ostatních zaměstnanců nebo klient se dokázal dovtípit, jaký výnos má společnost ze spolupráce s jinými klienty). V rámci IS je autorizace implementována ve formě ACL[30]. Klíčová je v tomto ohledu třída **Authorizator**, která se stará o zavedení dostupných oprávnění, rolí a pravidel pro ACL do Nette Frameworku. Všechny zmíněné záznamy jsou uchovávány v databázi, konkrétně v relacích **permission**, **role** a **role_permission**. ACL následně obsahuje záznamy na bázi uživatelských rolí. Nejprve dané uživatelské roli povolíme určité oprávnění a tuto roli následně nastavíme uživateli. Zavádění ACL do Frameworku Nette představuje operaci, která je prováděna při využití každé funkce IS. K minimalizaci počtu databázových dotazů si třída **Authorizator** uchovává nezbytné informace v cache a dotaz na databázi pokládá jen v případě cache miss[31], k čemuž dochází velmi ojediněle (např. po uložení

[30] ACL je zkratka pro anglický pojem „access control list“, do češtiny doslova překládáno jako „seznam pro řízení přístupu“. Jedná se o soubor pravidel, která definují, jaké operace smí daný uživatel (s danou rolí) provést.

[31] Termínem cache miss se označuje stav, kdy žádaná informace není v cache uložena.



Obrázek 7: Úkázka nástěnky

nového oprávnění do databáze a nutnosti smazat dosavadní cache).

3.6.1 Autorizace v praxi

Na mnoha místech zdrojového kódu potřebujeme rozhodnout, zda-li uživateli můžeme povolit provedení určité akce. K tomu postačí zavolat metodu **isAllowed** vyžadující jeden argument, který symbolizuje název prostředku, ke kterému chceme přistupovat. Všechny tyto prostředky jsou evidovány v databázi, konkrétně v relaci **permission** kompletně obsahující dostupná oprávnění, která můžeme přiřadit jednotlivým uživatelským rolím. Jméno role je složeno z jednotlivých logických částí oddělených dvojtečkou. Veškerá oprávnění týkající se statistik proto například začínají prefixem „Stats“.

3.7 Autentizace

K autentizaci dochází při každém přihlášení a potažmo i odhlášení uživatele. Jedná se tedy o proces, kdy pro účely IS zjišťujeme příslušnou identitu dané osoby. V aplikaci autentizaci obstarává třída **Authenticator**. Ta dokáže uživatele přihlásit dvěma způsoby – buď kombinací uživatelského jména a hesla nebo emailové adresy a hesla. Za zmínku dále stojí, že zmíněná třída obstarává výpočet

```php
<?php
/*Třída ExamplePresenter*/

//test, zda-li přihlášený uživatel smí
//přistupovat ke statistikám určeným pro klienta
$this->getUser()->isAllowed("Stats:Client");
?>
```

Zdrojový kód 10: Práce s metodou isAllowed

osoleného [32] hashe[33] hesla.

Po přihlášení jsou informace o aktuálním uživateli evidovány v objektu třídy **UserModel**. Tato třída dědí z třídy **Nette\Security\User**, kterou ve výchozím stavu používá Nette Framework. Soubor zmíněných informací framework označuje jako identitu. Ta obsahuje například přiřazenou uživatelskou roli a jako členskou proměnnou instanci třídy (entity) **User**. Třída **User** přitom eviduje veškeré informace o uživateli, které se nachází v databázi. Díky tomuto konceptu můžeme velmi jednoduše pracovat se zmíněnými objekty.

```php
<?php
/*Třída ExamplePresenter*/

//zjišťujeme, zda-li je uživatel přihlášený
$this->getUser()->isLoggedIn();
//zjištění emailové adresy aktuálně přihlášeného uživatele
$this->getUser()->getData()->email;
?>
```

Zdrojový kód 11: Práce s objektem přihlášeného uživatele

[32] Solení hesla představuje techniku, která případnému útočníkovi znesnadňuje získání předlohy (hesla) z otisku (hashe) vypočteného s pomocí jednosměrné hashovací funkce. Pokud útočník získá neoprávněný přístup do databáze, může manipulovat s hashem hesla konkrétního uživatele. I když výpočet předlohy není v tomto případě triviální (což zajišťuje právě ona zmíněná jednosměrná hashovací funkce), existují metody (útok hrubou silou (brute force attack), duhové tabulky (rainbow tables)) zaměřující se na nalezení předlohy. Výpočetní a časovou náročnost v případě útoku hrubou silou ovlivňuje výběr hashovací funkce, použití duhové tabulky ovšem dokáže zajistit nalezení předlohy (z množiny předem vypočítaných hodnot) v kratším čase oproti útoku hrubou silou. V případě IS je vstup od uživatele sloučen s textovým řetězcem (sůl), který se skládá z náhodně vygenerovaných znaků. Díky tomu existuje velmi malá šance, že by se výsledný hash nacházel v již existujících duhových tabulkách (útočník by totiž k jejich konstrukci musel znát použitou sůl).

[33] Termín hash symbolizuje výstup hashovací funkce, do češtiny jej lze přeložit jako „otisk“. V případě IS je využívána hashovací funkce SHA-2, konkrétně verze SHA-512. Oproti procesu šifrování se hashovaní liší mimo jiné tím, že je v rozumném čase prakticky nemožné zpětně rekonstruovat vstupní předlohu.

3.8 Cache

Během používání cache informační systém úzce spolupracuje s Nette Frameworkem. Vstupní bránu pro využití mezipaměti symbolizuje třída **AppCache** disponující několika statickými metodami, které se starají o konstrukci objektů třídy **AppCacheWorker**. Třída **AppCacheWorker** již slouží k samotné práci s cache. Objekt zmíněné třídy si pamatuje své interní označení, podle něhož si zvolí místo v rámci souborového systému, kde bude ukládat potřebná data. Každá instance disponuje metodou **load** načítající data z mezimapěti podle klíče předaného v argumentu. K uložení dat slouží metoda **save**.

```
1  <?php
2  public function loadPermissions() {
3     $cache = AppCache::systemCache();
4     $load = $cache->load("klic");
5
6     if ($load) {
7       //pakliže cache obsahuje data, vrátíme je
8       return $load;
9     } else {
10      //pokud ne, provede jejich získání
11      //(zpravidla výpočetně náročnější)
12      $load = ...
13      //data uložíme do cache
14      $cache->save("klic", $load);
15      return $load;
16    }
17 }
18 ?>
```

Zdrojový kód 12: Použití cache

V praxi vyvstala nutnost evidovat na centrálním místě skupiny dat ukládaných do cache. Programátor tak nemusí složitě procházet zdrojový kód, protože potřebný přehled získá na jediném místě. K tomuto účelu byla třída **AppCache** rozšířena o řadu textových konstant, které se používají jako klíče pro přístup k datům nacházejícím se v cache.

3.9 Emailové zprávy

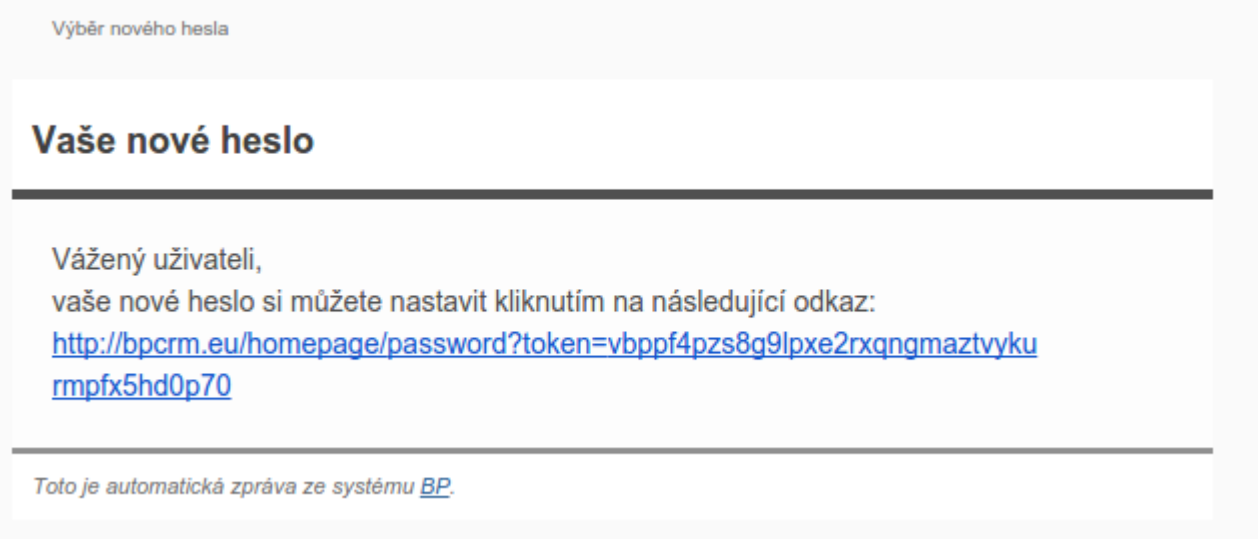
Implementované řešení pro zasílání emailových zpráv si klade za cíl eliminovat situaci, kdy by se na různých místech zdrojového kódu nacházely texty jednotlivých emailů. Sjednocením předpřipravených šablon do jednoho centrálního místa získáme rychlý přehled nad všemi dostupnými zprávami, které IS uživatelům zasílá. Z tohoto důvodu byla vytvořena třída **Mailer** obsahující základní metody pro přípravu objektu emailové zprávy. Tato třída úzce spolupracuje se šablonovacím systémem Latte a využívá tak všech jeho výhod. Z třídy **Mailer** děděním

vychází **MailerMember**, což je třída s konkrétní implementací nezbytnou pro zasílání zpráv uživatelům. Jako samostatné metody se zde nachází jednotlivé typy emailových zpráv, kterým v argumentu předáváme konkrétní údaje, jenž jsou doplňovány na místa proměnných v šablonách. Po zavolání metody tedy dojde k vygenerování výsledného HTML (s pomocí šablonovacího systému Latte) a vytvoření objektu emailové zprávy, který již jen čeká na své odeslání. K zefektivnění popsaného konceptu byla do procesu vytvoření emailu zapojena třída uživatelské entity **User**. K její instanci máme přístup napříč celým systémem a nese informaci o aktuálním uživateli. Po rozšíření třídy **User** o metodu **sendMeEmail** nám ji ve zdrojovém kódu stačí zavolat a jako povinný argument uvést interní textový řetězec určený pro rozpoznání zprávy, jenž budeme posílat. Pakliže to zvolená šablona vyžaduje, předáme ve volitelném argumentu asociativní pole obsahující hodnoty, které budou v emailu doplněny na určená dynamicky se obměňující místa. Metoda **sendMeEmail** si obstará instanci třídy **MailerMember** a zavolá patřičnou metodu generující tělo emailu, jenž vrátí objekty třídy **Message** dědící z **Nette\Mail\Message**, což je odpověď Nette Frameworku na potřebu komfortního zasílání emailových zpráv. V metodě **sendMeEmail** tedy nyní disponujeme objektem zprávy, který obsahuje nastavené tělo, přijemce a odesílatele zprávy, předmět i všechny ostatní náležitosti. Nám už pouze stačí zavolat finální metodu **send()** nevyžadující žádný argument, která se prostřednictvím nativní funkce jazyka PHP postará o zaslání zprávy.

Shrňme si, jaké důsledky s sebou implementované řešení přináší. Pokud vyvstane potřeba zasílání nové emailové zprávy, je na programátorovi, aby připravil šablonu a do třídy **MailerMember** dopsal novou metodu konkretizující daný email. Jakmile se rozhodneme vytvořenou zprávu odeslat, plně postačí zavolání jediné metody **sendMeEmail**.

```
1  <?php
2  /*Třída ExamplePresenter*/
3
4  $this->getUser()->getData()->sendMeEmail("welcome",
5                                array("username" => "user123"));
6  // Zasíláme zprávu interně identifikovanou jako "welcome",
7  // typicky tedy může jít např. o přivítání nového uživatele.
8  // Zmíněná šablona vyžaduje předání jedné proměnné,
9  // a to specifikování uživatelského jména.
10 // Proto na místě druhého argumentu spatříme
11 // asociativní pole obsahující kýžený údaj.
12 ?>
```

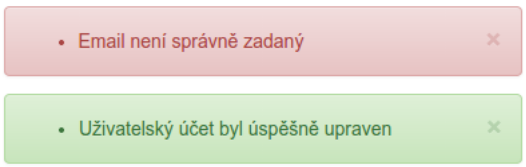
Zdrojový kód 13: Zasílání emailové zprávy



Obrázek 8: Ukázka zaslané emailové zprávy. V tomto případě se jedná o šablonu určenou pro nastavení zapomenutého hesla. Dynamicky je do těla textu doplňován unikátní odkaz.

3.10 Informační hlášky

V prostředí IS nejednou potřebujeme uživateli sdělit stručnou informaci. Častokrát jde o potvrzení jím provedené akce nebo vysvětlení, proč daný úkon nemůže realizovat. Popsaná zpráva se v terminologii Nette Frameworku označuje jako „flash message“.



Obrázek 9: Ukázka informační zprávy

Během vývoje byla vynaložena snaha na co nejjednodušší zobrazování dotyčných informačních zpráv, aby jejich použití kladlo na programátora co nejmenší nároky. Základ implementovanému systému poskytl Nette Framework, který disponuje metodou **flashMessage**, která se stará o zaregistrování dotyčné zprávy, ne však o její zobrazení. Zmíněná metoda přebírá v rámci argumentů text zprávy a její typ (např. zda-li se jedná o potvrzení, obyčejné konstatování nebo hlášku informující o chybě). Všechny zprávy připravené k zobrazení předává framework svému šablonovacímu systému. O jejich vykreslení se stará vytvořená komponenta (3.4), která podle typu zprávy rozhoduje o barevném podání (potvrzující zpráva je proto zelená, informační modrá a hlášení o chybě se zobrazí červeně).

Objekt komponenty následně stačí vykreslit v rodičovské šabloně, která ve svém těle zobrazuje všechny ostatní šablony. Tím je zajištěno, že informační hlášení můžeme v systému použít na libovolném místě bez nutnosti provádět změny v kódu platné jen pro určitou sekci IS. V konečném důsledku proto zobrazení hlášky probíhá zavoláním jediné metody.

```
1 <?php
2 /*Třída ExamplePresenter*/
3
4 //zobrazí hlášku potvrzující upravení uživatelského účtu
5 $this->flashMessage("USER/USER_EDIT", \BP\FlashMessages::GOOD);
6 ?>
```

Zdrojový kód 14: Vyvolání informační hlášky

Kvůli možnosti překladu aplikace do cizího jazyka (3.3) přebírá metoda **flashMessage** klíč definovaného textového řetězce, který je po překladu v šabloně zobrazen v korektním jazyce.

3.11 Formuláře

Každý formulář zobrazovaný v rámci IS je definován jako samostatná třída. Ty se nachází ve složce **app/forms**. Každý formulář má společného předka, kterým je třída **BaseForm** přinášející abstraktní metodu **buildForm** volanou po vytvoření instance každého konkrétního formuláře. V metodě **buildForm** definujeme jednotlivá vstupní pole formuláře (tato funkcionalita je součástí Nette Frameworku). Každý prvek formuláře lze při jeho vytváření obohatit o příslušné validátory i výchozí hodnoty. Finální implementace třídy prezentující určitý formulář proto obsahuje vše nezbytné pro vykreslení a validaci vstupních dat.

```
1  <?php
2  class FulltextSeach extends BaseForm
3  {
4    protected function buildForm()
5    {
6      $this->addText("fulltext", 'COMMON/FULLTEXT_SEARCH')
7            ->setAttribute('placeholder', 'COMMON/FULLTEXT_SEARCH');
8
9        $this->addSubmit("submit", 'COMMON/FULLTEXT_BUTTON_SEARCH');
10   }
11 }
12 ?>
```

Zdrojový kód 15: Implementace formuláře, zde konkrétně třída pro fulltextové vyhledávání

V uvedeném příkladě (zdrojový kód 15) vytváříme formulář obsahující 2 prvky. Vstupní textové pole (přidané metodou **addText**) a odesílací tlačítko (přídává **addSubmit**). Vždy musíme specifikovat jméno daného prvku, pod kterým budeme moci následně přistoupit ke konkrétní hodnotě vyplněné uživatelem. Dále udáváme popisek prvku. Kvůli možnosti překladu do cizího jazyka (3.3) i zde uvádíme textový řetězec, jenž IS interně zpracuje a nahradí konkrétním překladem. S prvkem lze dále pracovat, v uvedeném příkladu (kód 15) specifikujeme hodnotu atributu **placeholder**[34].

Obrázek 10: Vykreslený formulář

3.11.1 Vlastní validátory

Nette Framework obsahuje řadu základních užitečných validátorů vstupních dat. Ty po uživateli mohou například vyžadovat splnění určité minimální délky vstupu nebo uvedení celočíselné hodnoty. Pro kontrolu složitějších podmínek bylo nutno vytvořit sadu pravidel, jenž se nachází ve třídě **BaseForm**. S jejich pomocí například zajistíme, že vstupní pole nedovolí zadání uživatelského jména, které by nebylo v rámci IS unikátní.

```
1  <?php
2  abstract class BaseForm extends \Nette\Application\UI\Form
3  {
4    //konstanta pro identifikaci validátoru
5    const UNIQUE_USERNAME = "BP\BaseForm::validateUniqueUsername";
6
7    public static function validateUniqueUsername($item)
8    {
9      //získáme repozitář pro práci s uživateli
10       $repository = \Helper::getUserRepository();
11       //najdeme záznam podle uživatelského jména
12       $row = $repository->getByUsername($item->value);
13
14       //porovnáme návratovou hodnotu
15       return $row === FALSE;
16   }
17 }
```

Zdrojový kód 16: Definice vlastního validátoru, zde konkrétně kontrola unikátnosti uživatelského jména

[34]Placeholder je atribut pro použití v rámci formulářového vstupního pole v prostředí HTML. Jde o ukázkovou hodnotu, která uživateli udává, jaký typ údaje formulář očekává.

```php
<?php

class UserModifyForm extends BaseForm
{
  protected function buildForm()
    {
      //vytvoření vstupního textového pole,
      //které dovoluje zadání pouze unikátního
      //uživatelského jména
      $this->addText("username", "USERS/USERNAME")
      ->addRule(self::UNIQUE_USERNAME, "USERS/USERNAME_NOT_UNIQUE");
    }
}
?>
```

Zdrojový kód 17: Použití validátoru na kontrolu uživatelského jména

3.11.2 Zpracování formuláře

Třída formuláře popisuje jeho obsah a validační pravidla, neříká však nic o samotném zpracování zadaných dat. K této činnosti dochází v rámci presenteru. Zde vytvoříme novou komponentu (3.4) vracející instanci třídy námi vytvořeného formuláře. S pomocí callbacků[35] definujeme, jak se má IS zachovat v případě chybného a korektního zadání dat. V první zmíněné situaci proto uživateli s pomocí informačních zpráv (3.10) vysvětlíme, jaká pole vyžadují úpravu. Po jejich provedení a úspěšném odeslání formuláře již dochází k samotnému zpracování.

[35]Callback je označení pro kód, který je předáván jako argument jiné metodě či funkci s tím, že později dojde k jeho spuštění.

```
1  <?php
2  /*Třída ExamplePresenter*/
3
4   public function createComponentModifyForm() {
5
6      // vytvoříme novou instanci formuláře
7      $f = new \BP\UserModifyForm();
8
9      // po úspěšném odeslání formuláře zavoláme metodu doUserModify
10     $f->onSuccess[] = callback($this, 'doUserModify');
11
12     // neúspěšné odeslání formuláře vyvolá metodu doFormError,
13     // která uživateli zobrazí důvody chybného vstupu
14     $f->onError[] = callback($this, 'doFormError');
15     return $f;
16     }
17 ?>
```

Zdrojový kód 18: Vytvoření instance formuláře

4 Uživatelská příručka

Vyvinutý IS slouží ke správě a organizaci základních vnitrofiremních procesů IT společnosti. Namísto prostého popisu jednotlivých funkcí aplikace následuje popis nejčastějších úkonů, které budou chtít uživatelé v rámci IS vykonávat.

4.1 Administrátor

Uživatelský účet s administrátorskými právy v praxi zastupuje vedení společnosti nebo vrcholový management. Administrátor disponuje přístupem ke všem interním informacím a statistikám. Mezi základní úkoly administrátora patří vytváření nových uživatelských účtů, a to buď s rolí klienta nebo zaměstnance. Administrátor sleduje stav prací na jednotlivých projektech a zajímá se o předpokládaný zisk. Dále získává podklady pro provedení základních finančních operací firmy (vytvoření nové faktury s pomocí integrovaného fakturačního systému, výplata mezd zaměstnancům).

4.2 Zaměstnanec

Uživatel s právy zaměstnance využívá v rámci IS modul, který udržuje a organizuje úkoly čekající na zpracování.

4.3 Klient

Klient na IS spoléhá ve chvílích organizace a efektivního vedení zadaných požadavků. Dále pracuje s vystavenými fakturami.

4.4 Jak upravím fakturu?

V rámci zachování konzistentních podkladů pro vedení účetnictví není možné editovat již vystavenou fakturu. Nicméně tuto fakturu lze stornovat, čímž je umožněno zpětně editovat libovolnou položku zahrnutou v původní faktuře. Po úpravě dat stačí s pomocí integrovaného průvodce vystavit novou fakturu.

4.5 Jak přidám zaměstnance nebo klienta?

Každý uživatel musí mít v systému vytvořený svůj základní účet. Ten vytvoří administrátor v sekci Administrace → Uživatelé. Po zadání uživatelského jména, emailové adresy, jména i příjmení dané osoby a výběru role je na udanou adresu odeslána zpráva s potvrzovacím odkazem, s jehož pomocí si uživatel nastaví své heslo. V tuto chvíli se již uživatel může do systému přihlásit, prakticky však nemůže provést žádnou operaci, protože IS v tuto chvíli nevede o uživateli jeho zaměstnanecké nebo klientské informace.

4.5.1 Zaměstnanec

V sekci Administrace → Zaměstnanci čeká na administrátora úkon v přidání nového zaměstnance. Během vyplňování požadovaných informací je nutno vybrat základní uživatelský účet, čímž dojde ke spárování se zaměstnaneckými informacemi.

4.5.2 Klient

Zavedení klienta do systému vyžaduje návštěvu sekce Administrace → Klienti. Během vytváření nového klienta je třeba vybrat alespoň jeden základní uživatelský účet, ke kterému budou klientské informace přiřazeny. Následně je vhodné uložit do IS všechny projekty, které budou pro klienta zpracovávány. To se děje v části Administrace → Projekty.

4.6 Jak přiřadím zaměstnance k projektu?

Cheme-li přiřadit zaměstnance k vybranému projektu, je nutné, aby v systému již byl evidován daný zaměstnanec a vybraný projekt. Následně v sekci Administrace → Projekty postačí zobrazit detail zvoleného projektu. Administrátorovi je následně umožněno přiřadit k projektu nového zaměstnance. Během tohoto úkonu dochází k vymezení hodinové odměny a fakturované částky.

4.7 Jak zapíšu odvedenou práci?

Následující úkon bývá mezi zaměstnanci prováděn nejčastěji. V sekci Vývoj → Práce je potřeba vybrat projekt, kterého se práce týkala a uvést, v jaké části konkrétního dne byla prováděna. Nedílnou součást vytvářeného zápisu symbolizuje i stručný ale výstižný popis zpracovaného požadavku. Pokud se vynaložený čas týkal jednoho konkrétního úkolu vedeného v rámci IS, je možno jej k záznamu přiřadit, čímž dojde ke vzájemnému provázání.

4.8 Jak zobrazím statistický přehled?

Veškeré dostupné přehledy jsou situovány v sekci Statistiky. Nejbohatší možnosti zde přitom má uživatel s právy administrátora, ostatní typy účtů jsou v této oblasti limitovány. Statistiky odráží nejen aktuální dění v obchodní společnosti, ale pamatují i na možnost zobrazení informací z minulosti.

4.9 Kde naleznu informace o výplatách?

Nárok na výplatu vzniká automaticky během generování faktury. Podle množství odvedené práce a specifikované hodinové mzdy daného zaměstnance je vypočtena suma, na kterou má daný pracovník nárok. Toto číslo lze vnímat jako podklad pro výpočet čisté mzdy, kdy pochopitelně závisí [2] na konkrétních sjednaných podmínkách pracovního poměru. Po samotné úhradě výplaty lze záznam v IS označit jako vyřešený.

4.10 Kde nastavím základní chování IS?

Přístup do sekce s nastavením je umožněn pouze uživateli s právy administrátora. Ten zde může snadno ovlivnit základní parametry, s nimiž IS operuje. Praxe ukázala, že klíčovou se stává potřeba bezproblémové změny výše DPH, a to pochopitelně kdykoliv během doby používání IS. Další položky v nastavení s největší pravděpodobností vedení společnosti specifikuje ve fázi prvotního nasazení IS a následně k jejich změnám nijak často docházet nebude. Jde o uvedení obchodních informací o dané firmě, které se zobrazují na vygenerovaných fakturách. IS umožňuje specifikovat prefix, jenž se objeví před číslem faktury, díky čemuž vzniká unikátní číselná řada, která nezasahuje do faktur vydaných ve společnosti před zavedením IS.

4.11 Jak vytvořím novou fakturu?

Vygenerování nové faktury je nezbytný úkon pro vytvoření nové pohledávky [36] a potažmo i závazků vůči zaměstnancům. V sekci Finance → Faktury se nachází

[36] Pod pojmem pohledávka [2] si lze představit právo na zaplacení peněz. Pokud např. firma A má pohledávku za firmou B, znamená to, že firma B dluží firmě A peníze, a tedy firma A požaduje od firmy B vyrovnání dluhu.

průvodce vytvořením nové faktury. Během jejího generování je třeba zvolit projekt a určit konec fakturačního období. Následně se zobrazí záznamy o odvedené práci, které odpovídají aplikovaným filtrům. Z dostupných možností následně administrátor vybírá ty položky, které se stanou součástí budoucí faktury. Po finálním potvrzení lze fakturu mimo jiné stáhnout ve formátu PDF.

4.12 Jak zjistím podrobnosti o zadaném požadavku?

Klienta i vedení společnosti bude častokrát zajímat aktuální stav práce na dílčím úkolu. Každému požadavku je během jeho vytvoření přiřazen unikátní číselný identifikátor, který jej jednoznačně reprezentuje a usnadňuje odkazování na daný požadavek během komunikace. Každý úkol má v IS vytvořenou vlastní stránku obsahující detaily týkající se požadavku. Kromě samotného zadání se zde nachází komentáře od ostatních uživatelů IS, které mohou požadavek dále upřesňovat.

4.13 Jak se rychle seznámím s aktuálním stavem a zjistím, co vyžaduje moji pozornost?

Každý uživatel spatří bezprostředně po svém přihlášení do systému personalizovanou nástěnku, která obsahuje informace vztahující se k aktuálnímu uživateli. Kategorie dostupných zpráv se liší podle uživatelské role. IS vždy do výpisu zahrne jen ty kategorie, které aktuálně chtějí uživateli sdělit relevantní informaci. Vybrané kategorie jsou následně náhodně promíchány, díky čemuž nástěnka ztrácí na své jednotvárnosti a může uživatele více motivovat k přečtení jednotlivých zpráv.

4.13.1 Zaměstnanec

Praxe ukázala, že zaměstnanec z pochopitelných důvodů vyžaduje rychlý přístup k úkolům, které aktuálně zpracovává. Právě proto nástěnka zobrazuje vybraný počet požadavků přiřazených danému pracovníkovi. Jakmile je v systému zaevidována nová dosud nevyplacená výplata, je tato informace rovněž na nástěnce reflektována.

4.13.2 Klient

Zákazník společnosti využívající IS podobně jako její zaměstnanec ocení rychlý přístup k aktuálním úkolům. Oproti zaměstnanci však motivace vychází z přání rychle se seznámit s aktuálním stavem prací a zodpovědět případné dotazy. Nedílnou součástí obchodního vztahu s dodavatelskou firmou je evidence vystavených faktur. Na ně IS rovněž upozorňuje prostřednictvím zmiňované nástěnky. Ze zřejmých důvodů je zvláštní péče věnována dosud neproplaceným fakturám.

4.13.3 Administrátor

Vedení společnosti vyžaduje rychlý přístup k několika druhům informací. Podobně jako klient, i administrátor vyhledává podrobnosti o dosud nezaplacených fakturách, aby ze strany firmy nebyla v ohrožení schopnost dostát svým závazkům. U zaměstnanců, u nichž je vedena doba jejich smluvního závazku nebo období, kdy vystupují v roli studenta, provádí IS kontrolu blízké expirace zmíněné doby. Administrátor je na nutnost prodloužení smluvního závazku nebo vyžádání potvrzení o studiu informován s dostatečným časovým předstihem. Vedení společnosti je dále upozorňováno na dosud neproplacené mzdy. Efektivní řízení firmy vyžaduje dokončování zadaných úkolů v dohodnutém čase. Proto nástěnka zobrazuje vybrané nedokončené požadavky, na kterých se aktuálně pracuje. Administrátor se tak rychle seznámí s aktuálním stavem a pokud to situace vyžaduje, provede opatření s cílem minimalizovat prodlužování doby pro zpracování úkolu.

4.14 Jak jsem informován o dění v rámci IS, i když nejsem přihlášen?

Po přihlášení do IS uživatel rychlé získává přehled o aktuální situaci a nachází zde odpovědi na své otázky. O některých událostech by však měl být uživatel informován i v době, kdy není do systému aktuálně přihlášen. Právě proto IS rozesílá informační email při každé podstatné operaci týkající se pracovního úkolu. V praxi se totiž ukázalo, že právě efektivní komunikace mezi zadavatelem a osobou zpracovávající daný úkol dokáže zkrátit dobu potřebnou pro splnění požadavku. Typický scénář se nese v duchu odpovědi na otázku týkající se konkrétního úkolu. Zmíněný dotaz může pocházet jak od klienta, tak i pracovníka. Zúčastněná strana je přitom bezprostředně informována právě prostřednictvím emailové zprávy. Po přihlášení do IS a zaznamenání odpovědi se o ní shodně pomocí emailu dozvídá původní tazatel.

4.15 Co mohu dělat, když zapomenu heslo?

Pakliže uživatel zapomene své přihlašovací heslo, existuje jednoduchý postup, jak nastalou situaci vyřešit. Na úvodní stránce IS je připraven formulář pro opětovné nastavení zapomenutého hesla. Po uživateli je vyžadováno zadání registrované emailové adresy, na kterou dorazí zpráva s unikátním odkazem. Po jeho navštívení je uživateli prezentován formulář, ve kterém specifikuje své nové heslo, s nímž se následně může přihlásit.

Závěr

Vyvinutý informační systém nabízí dostatek nástrojů pro nasazení v prostředí obchodní společnosti podnikající v oblasti IT. Spojuje vedení společnosti, zaměstnance i zákazníky, přičemž myslí na specifické potřeby uživatelů jednotlivých skupin.

Aplikace uchovává údaje o jednotlivých požadavcích zákazníků. Na nich následně pracují zaměstnanci a evidují strávenou dobu. V případě nutnosti mohou jednoduše kontaktovat samotného klienta, který osvětlí případné nejasnosti týkající se dílčího požadavku. Na základě odpracované doby systém vytváří podklady pro faktury a vyplácení mezd.

Povaha vedených informací dovoluje generování různorodých statistických přehledů. S jejich pomocí se vedení společnosti snadněji seznamuje s aktuálním stavem práce na jednotlivých projektech a s větší jistotou plánuje budoucí aktivity firmy.

Oproti původním představám nabízí informační systém i některé funkce navíc, jejichž užitečnost vyvstala během vývoje aplikace. Jedná se o správu zaměstnaneckých poměrů s jednotlivými pracovníky, s cílem zpřehlednit nezbytnou administrativu. Systém eviduje jednotlivé smluvní závazky a s dostatečným časovým předstihem upozorňuje na jejich případné vypršení.

Budoucí vývoj informačního systému se ponese v duchu zakomponování nových funkcí, jejichž potřebu ukáže dlouhodobější nasazení v praxi. Aktuálně se jmenovitě jedná o modul schopný vytvářet nové požadavky na základě doručených emailových zpráv od zákazníků. Aplikace bude v pravidelných intervalech kontrolovat přidělenou emailovu schránku a každou novou zprávu zaeviduje do systému, kde se jí následně bude zabývat pověřený pracovník.

A Příloha: Návod ke spuštění aplikace

Minimální požadavky vyvinutého IS jsou dány především použitými knihovnami. Pro svůj bezproblémový běh proto aplikace očekává splnění následujících podmínek.

Na straně serveru vyžadujeme:

- webový server (např. Apache ve verzi 2)
- podporu PHP 5, konkrétně minimálně verzi 5.4
- databázový server MySQL, a to ve verzi 5.5 nebo vyšší
- splnění nutných podmínek, které s sebou nese použití Nette Frameworku
  - k ověření této závislosti je možno využít specializovaný nástroj označovaný jako „Requirements checker“. Ten se na přiloženém CD nachází ve složce `install/checker/`. Obsah této složky postačí nakopírovat do vybraného umístění webového serveru a následně obsah složky zobrazit ve webovém prohlížeči. Zobrazí se přehledná tabulka informující o splnění jednotlivých závislostí. Bližší informace o tomto nástroji lze nalézt na http://doc.nette.org/cs/2.2/requirements.
- korektní konfiguraci poštovního serveru, který spolupracuje s PHP 5. K odesílání zpráv je využívána funkce **mail** (http://php.net/manual/en/function.mail.php)

Pro použití IS na straně klienta vyžadujeme:

- moderní webový prohlížeč (tedy webový prohlížeč ve verzi, kterou jeho vydavatel neoznačil za zastaralou)
  - povolený JavaScript
  - povolené HTTP cookies

A.1 Instalace aplikace

A.1.1 Databáze

IS ukládá data, s nimiž pracuje, do relační databáze. Na přiloženém CD je ve složce `install/dump/` připraven SQL dump databáze, který je nutno nahrát (naimportovat) v prostředí databázového serveru MySQL.

A.1.2 Webový server

1. Do zvolené složky webového serveru je třeba překopírovat veškerý obsah archívu, který se na přiloženém CD nachází ve složce `bin/`
2. Webovému serveru je třeba povolit zapisování do složek `temp/` a `log/` (ty vznikly překopírováním souborů v předcházejícím kroku).

3. Následuje nezbytná konfigurace aplikace. Je třeba specifikovat údaje potřebné pro připojení k databázovému serveru. V textovém editoru je nutno otevřít soubor `app/config/config.production.neon`, jehož jednotlivé řádky v komentářích uvádí očekávané hodnoty. Minimálně je třeba uvést pojmenování databáze v kombinaci s přihlašovacím jménem a heslem uživatele, pod nímž bude aplikace do databáze přistupovat.

Aplikace je po provedení výše zmíněných kroků připravena k použití. Ve webovém prohlížeči postačí otevřít platné umístění, v němž se nachází složka `www/`. Pakliže například pro soubory aplikace byla v kořenovém umístění lokálního webového serveru vytvořena složka `example`, zobrazí se úvodní stránka po zadání http://127.0.0.1/example/www/. K prvnímu přihlášení je připraven uživatelský účet s právy administrátora.

- uživatelské jméno: **admin**
- heslo: **admin.admin**

Přihlašovací údaje k dalším uživatelským účtům se nachází na CD v souboru `readme.txt`. Veřejná testovací verze IS je dostupná na http://bpcrm.eu/.

A.2 Řešení možných potíží

A.2.1 Problémy se spuštěním aplikace

Pakliže se první spuštění aplikace nezdaří, je pravděpodobné, že příčinu chyby framework zapsal do umístění `log/`. V této složce jsou vytvářeny samostatné soubory s koncovkou `.html`, které po otevření ve webovém prohlížeči přehledně seznamují s nastalou chybou.

A.2.2 Potíže s doručováním emailových zpráv na uvedenou adresu

Doručení emailové zprávy z prostředí aplikace do emailové schránky příjemce vyžaduje korektní konfiguraci poštovního serveru, který bude v procesu doručování zprávy vyhodnocen jako důvěryhodný. Populární bezplatné emailové služby v rámci boje proti příjmu nevyžádané pošty posuzují zaslanou zprávu sadou pravidel, po jejichž vyhodnocení může dojít k přesunu do složky s nevyžádanou poštou (tzv. spam) nebo naopak k doručení do schránky příjemce dojít nemusí a zpráva bude zahozena. Volitelně lze aplikaci zakázat odesílání emailových zpráv. V konfiguračním souboru `app/config/config.neon` postačí nastavit parametr **sendMails** na hodnotu **false**.

B Obsah přiloženého CD

`bin/`
: Složka obsahující archív s kompletní adresářovou strukturou webové aplikace. Tyto soubory aplikace vyžaduje pro svůj provoz na webovém serveru (A.1.2).

`doc/`
: Následující složka obsahuje text práce ve formátu PDF. Dále se v tomto umístění nachází archív obsahující veškeré soubory nezbytné pro vygenerování PDF dokumentu.

`src/`
: Kompletní zdrojové kódy webové aplikace. Zde umístěná adresářová struktura je nezbytná pro běh na webovém serveru.

`readme.txt`
: Tento soubor popisuje minimální požadavky nezbytné pro spuštění aplikace. Dále obsahuje instrukce pro nasazení aplikace do prostředí webového serveru.

Navíc CD obsahuje:

`install/`
: Tato složka obsahuje dump SQL databáze, který je nezbytný pro spuštění aplikace (A.1.1). Dále se zde nachází nástroj pro interaktivní ověření minimálních požadavků použitého Nette Frameworku.

Literatura

[1] KOTÍKOVÁ, Halina a Jaroslav ZLÁMAL. *Základy marketingu*. 1. vyd. Olomouc: Univerzita Palackého v Olomouci, 2006, 77 s. ISBN 80-244-1489-9.

[2] OPLETALOVÁ, Alena. *Základy účetnictví*. 1. vyd. Olomouc: Univerzita Palackého, 2006, 88 s. ISBN 80-244-1296-9.

[3] *Nette Foundation* [online]. © 2008-2014 [cit. 2014-11-06]. Dostupné z: http://nettefoundation.com/

[4] *Nette Framework* [online]. © 2008-2014 [cit. 2014-11-06]. Dostupné z: http://nette.org/

[5] *Bootstrap* [online]. © 2014 [cit. 2014-11-06]. Dostupné z: http://getbootstrap.com/

[6] Word 2007 HTML and CSS Rendering Capabilities in Outlook 2007 (Part 1 of 2). *Microsoft Developer Network* [online]. © 2014 [cit. 2014-11-06]. Dostupné z: http://msdn.microsoft.com/en-us/library/aa338201(v=office.12).aspx

[7] mailchimp/email-blueprints. *GitHub* [online]. © 2015 [cit. 2015-01-31]. Dostupné z: https://github.com/mailchimp/Email-Blueprints

[8] *Email Client Market Share* [online]. c 2014 [cit. 2014-11-06]. Dostupné z: http://emailclientmarketshare.com/

[9] *Lean Mapper* [online]. © 2013 [cit. 2014-11-18]. Dostupné z: http://www.leanmapper.com/

[10] OndrejBrejla/Eciovni. *GitHub* [online]. © 2014 [cit. 2014-11-18]. Dostupné z: https://github.com/OndrejBrejla/Eciovni

[11] *JQuery* [online]. © 2015 [cit. 2015-01-16]. Dostupné z: http://jquery.com/

[12] *MPDF* [online]. © 2005 - 2013 [cit. 2015-01-16]. Dostupné z: http://www.mpdf1.com/mpdf/index.php

[13] *Select2 - The jQuery replacement for select boxes* [online]. © 2015 [cit. 2015-01-17]. Dostupné z: https://select2.github.io/

[14] *Bootstrap datetimepicker* [online]. © 2015 [cit. 2015-01-17]. Dostupné z: http://eonasdan.github.io/bootstrap-datetimepicker/

[15] *Flot: Attractive JavaScript plotting for jQuery* [online]. © 2007 - 2014 [cit. 2015-01-17]. Dostupné z: http://www.flotcharts.org/