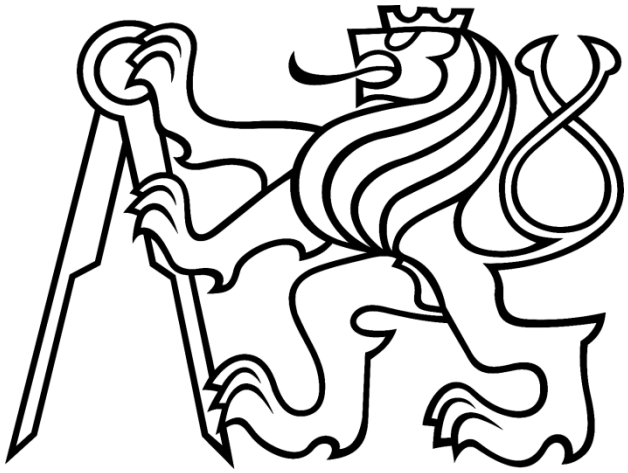
České vysoké učení technické v Praze
Fakulta elektrotechnická



Bakalářská práce

Call centrum pro průzkum spokojenosti zákazníků

Martin Dendis

Vedoucí práce: Ing. Martin Klíma, Ph.D.

Studijní program: Otevřená informatika, Bakalářský

Obor: Informatika a počítačové vědy

23. května 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Martin D e n d i s

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Call centrum pro průzkum spokojenosti zákazníků

Pokyny pro vypracování:

Navrhněte a implementujte část funkcionality call centra pro průzkum spokojenosti zákazníků.
Aplikaci navrhněte jako webovou nad technologií PHP/JavaScript.
Aplikace umožní:

- stahování a přípravu kontaktů pro obvolávání,
- export nasbíraných dat pro další zpracování,
- zobrazení a případnou editaci dotazníku,
- editaci seznamu partnerů a jejich týdenních kvót,
- editaci seznamu robinsonů (kontakty, jež si již nepřejí být v budoucnu kontaktovány),
- editaci seznamu zakázaných kontaktů (zahraniční, placená aj. čísla).

Specificky se zaměřte na tyto oblasti:

- systém pro automatické stahování nových kontaktů,
- systém pro ruční import nových kontaktů pro obvolávání,
- filtrovací systém nových kontaktů,
- dotazovací systém call centra,
- systém pro generování reportů,
- generování souhrnných statistik z celého roku ve formátu IBM SPSS.

Práci koordinujte a spolupracujte s panem Filipem Šamánkem, který pracuje na podobném tématu v rámci své bakalářské práce.

Seznam odborné literatury:

[1] Matt Zandstra - PHP Objects, Patterns, and Practicem - November 26, 2013 - ISBN-10: 1430260319, ISBN-13: 978-1430260318.
[2] Baron Schwartz, Peter Zaitsev, Vadim Tkachenko – High Performance MySQL: Optimization, Backups, and Replication March 30, 2012 - ISBN-10: 1449314287, ISBN-13: 978-1449314286.
[3] Darren George, Paul Mallery - IBM SPSS Statistics 21 Step by Step: A Simple Guide and Reference (13th Edition) July 26, 2013 - ISBN-10: 0205985513, ISBN-13: 978-0205985517.

Vedoucí bakalářské práce: Ing. Martin Klíma, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

Poděkování

Rád bych zde poděkoval panu Ing. Martinu Klímovi, Ph.D. za jeho rady a čas, který mi věnoval při řešení dané problematiky. Dále bych rád poděkoval svým rodičům za jejich podporu během mého studia. V neposlední řadě bych pak chtěl poděkovat Filipu Šamánkovi, který v rámci vlastní bakalářské práce pracoval na obdobném tématu a s kterým jsem úzce spolupracoval.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ……. ……

Podpis autora práce

Abstrakt

Tato práce je věnována problematice call centra pro odchozí hovory, které se může využívat pro průzkum marketingu a spokojenosti zákazníků s produkty a poskytovanými službami různých společností. Práce se nejprve zabývá analýzou základních požadavků na call centrum. Na základě této analýzy je následně předložen návrh možného řešení a vlastní implementace v jazyce PHP. Dále práce obsahuje výsledky testování systému dotazníků call centra a vyhodnocení dosažených výsledků.

Abstract

This thesis is dedicated to the issue of call centre for out-band calls, which can be used for marketing and survey of customer satisfaction with the products and services provided by various companies. First, it analyses the essential requirements of the call centre. Based on this analysis, a solution and the implementation in PHP is proposed. The work also contains the results of testing of the questionnaire system of the call centre and evaluation of results.

Obsah

1. Úvod

Firma A poskytuje či zprostředkovává služby v rámci marketingu, výzkumu a monitoringu trhu mezinárodním společnostem.

Mimo jiné také Firma A zprostředkovává pro tyto společnosti osobní rozhovory zvané CATI[1] v podobě call center, které slouží pro výzkum trhu a spokojenosti zákazníků s produkty a službami poskytovanými těmito společnostmi.

Tato práce se zabývá návrhem a implementací aplikace pro CATI, jež bude použita pro výzkum spokojenosti zákazníků. Konkrétně bude prováděn výzkum spokojenosti zákazníků s autorizovanými partnery Firmy B.

Výsledky výzkumu pak budou předávány zpět do Firmy B v podobě různých XLS[2] reportů. Dále budou výstupem call centra reporty ve formátu SPSS[3], jež budou automaticky zasílány do Firmy C, která pak pro Firmu B vyhotoví podrobné statistiky jednotlivých partnerů, na jejichž základě bude Firma B zlepšovat své dosavadní služby.

Firma A předložila specifické požadavky pro výsledné call centrum Firmy B (dále jen CSS[4]). Detailní specifikace byly označeny jakožto soukromé informace, proto se tato práce bude zabývat pouze okrajovým popisem základních vlastností a funkcionality CSS, které by mělo implementovat každé takovéto call centrum. Rovněž výsledná implementace přiložená k této práci nebude přesně odpovídat finálnímu řešení, ale bude pouze obsahovat funkcionalitu popisovanou v této práci.

Aby nedošlo ke zkreslení při testování výsledné aplikace, testování proběhne na strojích CSS a za použití reálných dat, které byly shromažďovány po dobu jednoho roku provozu call centra, které dříve provádělo výzkum pro Firmu B.

1.1 Cíle práce

Cílem této práce je nastudovat obdržené podklady a specifikace CSS, navrhnout systém call centra, následně tento systém implementovat a otestovat jeho funkčnost. Na závěr bude provedeno zhodnocení celé aplikace, zdali odpovídá předloženým specifikacím a uvedeny návrhy na další možný rozvoj CSS.

[1] CATI - Computer-Assisted Telephone Interviewing
[2] XLS – datový formát aplikace Microsoft Office Excel
[3] SPSS - Statistical Package for the Social Sciences, formát společnosti IBM pro statistická data
[4] CSS - Customer Satisfaction Survey

2. Funkční požadavky

Nejprve budou představeny uživatelské role CSS a popis jejich pravomocí a úloh, které budou v call centru vykonávat.

Následně pak bude představeno základní schéma procesu zpracování informací v CSS s podrobným rozborem jednotlivých akcí a jejich požadavků.

V závěru rozboru funkčních požadavků pak budou detailněji rozebrány některé požadavky na práci s daty v CSS.

2.1 Uživatelské role

Dle obdržených specifikací budou k CSS přistupovat celkem tři hlavní role uživatelů. Konkrétně to jsou role *agent*, *supervizor* a *klient*. Každá z uvedených rolí sehrává specifickou a nenahraditelnou úlohu v CSS. Dohromady pak obstarávají celý chod call centra.

Navíc bude systém obsahovat speciální roli *support*, která bude sloužit pro snadný přístup servisního technika do aplikace CSS.

2.1.1 Agent

- Provádí osobní rozhovory se zákazníky Firmy B.
- Každý agent má vlastní frontu kontaktů, kterou mu systém automaticky doplňuje[5].
- Dále má každý agent přístup k vlastní historii provedených hovorů. Zde bude stačit zobrazit pouze výčet několika posledních hovorů, bez možnosti přístupu k vyplněnému dotazníku, či k audio nahrávce rozhovoru.
- Pokud agent při vyplňování dotazníku udělá chybu, nahlásí ji supervizorovi a ten provede korekci dotazníku (např. za pomoci audio nahrávky rozhovoru). Díky tomu bude mít supervizor i lepší přehled o odváděné práci jednotlivých agentů.
- Pro zahájení rozhovoru agent dle vlastního uvážení vybere z přidělené fronty jeden z kontaktů, přičemž dojde k navázání spojení se zákazníkem a agentovi bude zobrazen příslušný dotazník.
- Při provádění rozhovoru má agent možnost kdykoliv rozhovor ukončit. Dále má přístup k základním informacím o zákazníkovi, se kterým rozhovor provádí (např. jméno, telefon, datum návštěvy partnera Firmy B aj.).
- Po vyplnění dotazníku mu agent přiřadí stav (např. dokončeno, přeplánováno, omyl) a dotazník uloží[5].

[5] Podrobnější informace lze nalézt v bakalářské práci Filipa Šamánka

2.1.2 Supervizor

- Obstarává a řídí chod celého CSS.
- Provádí monitoring a správu agentů, k tomuto účelu má k dispozici následující prostředky[6]:
    1. Možnost online příposlechu hovoru.
    2. Zobrazení stavu agenta (např. provádí rozhovor, přestávka aj.).
    3. Má přístup k vyplněným dotazníkům a audio nahrávkám všech rozhovorů.
    4. V případě potřeby má prostředky k odhlášení agenta z ústředny.
    5. Má prostředky pro editaci agentů.
- Má přístup k online statistikám zpracovávané dávky[7], konkrétně jde o tyto statistiky[6]:
    1. Přehled o plnění plánů[8].
    2. Statistiky o zpracovaných kontaktech.
- Supervizor provádí ruční kontrolu všech vyplněných dotazníků a je-li to potřeba, doplní do dotazníku chybějící informace. Z těchto důvodů má rovněž přístup k audio nahrávce rozhovoru. Zkontrolovaný dotazník má pak možnost schválit, čímž dojde k jeho publikaci ve webovém rozhraní. Supervizor rovněž může provést korekci dotazníků na žádost klientů z Firmy B.
- Supervizor má rovněž přístup k některým automaticky generovaným reportům call centra. Tyto reporty pak příslušným způsobem zpracuje a případně publikuje ve webovém rozhraní pro pracovníky z Firmy B.
- V ojedinělých případech může dojít k tomu, že budou kontakty obdrženy i jinak, nežli standardní cestou (viz kapitola: 2.4 Služba pro výměnu dat). V takovémto případě supervizor za pomoci připraveného nástroje nahraje data do systému, který provede jejich kontrolu a připraví je k obvolání.
- Supervizor má rovněž možnost editovat seznam partnerů, upravovat plán partnerů, editovat seznam Robinsonů[9] a seznam zakázaných telefonních čísel.

[6] Podrobnější popis těchto prostředků lze nalézt v bakalářské práci Filipa Šamánka
[7] Dávka – viz kapitola: 2.2 Plánování dávek
[8] Plán – počet rozhovorů, které mají být v rámci jedné dávky provedeny pro konkrétního partnera
[9] Robinson – zákazník (kontakt), jež si nepřeje být kontaktován v rámci marketingu a CSS

2.1.3 Klient

- Slouží pro externí přístup klienta (pracovníka Firmy B) do aplikace CSS přes webové rozhraní.
- Klient má přístup k souborům, které zveřejnil na webovém rozhraní supervizor a má možnost tyto soubory mazat, či uploadovat soubory nové. Při uploadu má rovněž možnost nechat zaslat na vybrané emailové adresy notifikaci o nově přidaném souboru.
- Dále má možnost procházet vyplněné dotazníky, jež byly zkontrolovány supervizorem a má možnost poslechu audio nahrávky rozhovoru.
- V případě, že klient nebude spokojen s vyplněním dotazníku, nahlásí supervizorovi požadované změny a ten na jejich základě provede korekci.
- Každý klient má také možnost editovat vlastní účet (jméno a heslo). Klient s rozšířenými pravomocemi (admin webového rozhraní) má pak možnost editovat i ostatní klienty a jejich oprávnění.

2.1.4 Support

- Slouží pro externí přístup servisního technika do aplikace CSS.
- Má prostředky pro správu *agentů, supervizorů* a *klientů* aplikace.
- Má možnost přistupovat uploadovaným souborům a k automaticky generovaným reportům výhradně za účelem jejich úpravy na žádost *supervizora,* nebo *klienta* aplikace CSS.
- Má schopnost ručně invokovat automatizované procesy CSS (např. za účelem přegenerování určitých reportů po dodatečné korekci vyplněných dotazníků) na žádost *supervizora,* nebo *klienta* aplikace CSS.
- Má schopnost přeplánovat automatizované procesy aplikace CSS.

2.2 Plánování dávek

Průběh zpracování kontaktů a chod CSS je řízen specifickým harmonogramem. Základní vlastností je, že se kontakty zpracovávají po dávkách. Dávka je pro zjednodušení ohraničena vždy jedním kalendářním týdnem, ve kterém zákazník navštívil autorizovaného partnera Firmy B. V následujícím týdnu jsou kontakty z dávky automaticky staženy aplikací CSS a následně proběhne jejich čištění (viz kapitola: 2.6 Čištění nových kontaktů). V dalším týdnu pak probíhá telefonické kontaktování jednotlivých zákazníků.

Pro účely měsíčního reportingu budou jednotlivé dávky zařazeny do konkrétního měsíce celé, tj. nepůjde o kalendářní měsíc, ale o upravený měsíc na celé týdny (viz Tabulka 1).

dávka	měsíc	sběr kontaktů pro CSS	příjem kontaktů	čištění kontaktů	zaslání statistik z čištění kontaktů Firmě B	obvolávání kontaktů	generování statistik týdně	generování statistik měsíčně	generování SPSS statistik pro Firmu C
		po-ne	pondělí	út-čt	pátek	po-pá	ne	ne	denně
201401	leden	30.12-05.01	06.01	07.01-09.01	10.01	13.01-17.01	19.01		
201402	leden	06.01-12.01	13.01	14.01-16.01	17.01	20.01-24.01	26.01		
201403	leden	13.01-19.01	20.01	21.01-23.01	24.01	27.01-31.01	02.02		
201404	leden	20.01-26.01	27.01	28.01-30.01	31.01	03.02-07.02	09.02		
201405	leden	27.01-02.02	03.02	04.02-06.02	07.02	10.02-14.02	16.02	16.02	
201406	únor	03.02-09.02	10.02	11.02-13.02	14.02	17.02-21.02	23.02		
201407	únor	10.02-16.02	17.02	18.02-20.02	21.02	24.02-28.02	02.03		
201408	únor	17.02-23.02	24.02	25.02-27.02	28.02	03.03-07.03	09.03		
201409	únor	24.02-02.03	03.03	04.03-06.03	07.03	10.03-14.03	16.03	16.03	
:	:	:	:	:	:	:	:	:	:
201445	listopad	03.11-09.11	10.11	11.11-13.11	14.11	17.11-21.11	23.11		
201446	listopad	10.11-16.11	17.11	18.11-20.11	21.11	24.11-28.11	30.11		
201447	listopad	17.11-23.11	24.11	25.11-27.11	28.11	01.12-05.12	07.12		
201448	listopad	24.11-30.11	01.12	02.12-04.12	05.12	08.12-12.12	14.12	14.12	
201449	prosinec	01.12-07.12	08.12	09.12-11.12	12.12	15.12-19.12	21.12		
201450	prosinec	08.12-14.12	29.12	30.12-01.01	02.01	05.01-09.01	11.01		
201451	prosinec	15.12-21.12	29.12	30.12-01.01	02.01	05.01-09.01	11.01		
201452	prosinec	22.12-28.12	29.12	30.12-01.01	02.01	05.01-09.01	11.01	11.01	

Tabulka 1- Vzorový harmonogram pro rok 2014[10]

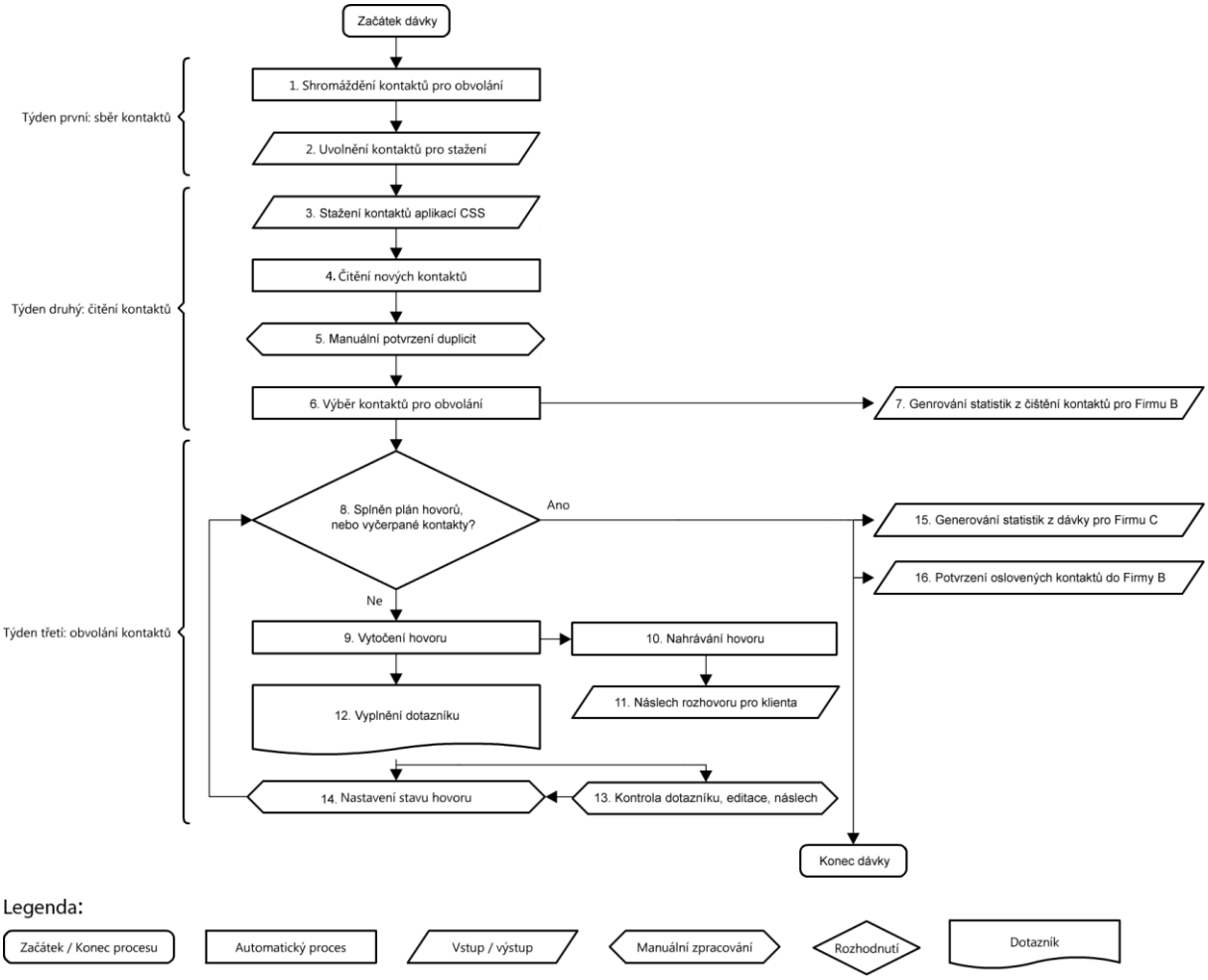
[10] Tabulka 1 slouží pouze pro lepší demonstraci způsobu rozložení dávek napříč kalendářním rokem a neodpovídá finálnímu plánování jednotlivých dávek

2.3 Schéma procesu zpracování informací v CSS

Následující schéma naznačuje základní kroky procesu zpracování informací v CSS. Tento proces lze rozdělit na tři po sobě jdoucí fáze (*sběr kontaktů*, *čištění kontaktů* a *obvolávání kontaktů*), kde se pak každá fáze zpracovává vždy podobu jednoho týdne.

Těchto procesů může být (a také většinou je) zpracováváno najednou více, maximálně však tři zároveň, kdy první proces je ve fázi *sběr kontaktů*, druhý proces ve fázi *čištění kontaktů* a třetí proces ve fázi *obvolávání kontaktů*. Jednotlivé procesy pak přechází do další fáze vždy na konci týdne.

Tento způsob zpracování lépe demonstruje přiložený Obrázek 1.



Obrázek 1 - Schéma procesu zpracování informací v CSS

2.3.1 Popis jednotlivých kroků procesu

1. ***Shromáždění kontaktů pro obvolání***

Firma B na konci týdne shromáždí od svých partnerů nové kontaktní údaje zákazníků, kteří jej za sledované období navštívili.

2. ***Uvolnění kontaktů pro stažení***

Po shromáždění těchto kontaktů budou kontakty k dispozici ke stažení přes *Službu pro výměnu dat* (viz kapitola: 2.4 Služba pro výměnu dat).

3. ***Stažení kontaktů aplikací CSS***

Aplikace CSS si bude aktivně stahovat nové kontakty prostřednictvím *Služby pro výměnu dat*.

4. ***Čištění nových kontaktů***

Nově stažené kontakty budou čištěny a některé údaje validovány na základě pravidel obdržených od Firmy B (viz kapitola: 2.6 Čištění nových kontaktů). Cílem je např. zajistit, aby jeden zákazník byl kontaktován maximálně jednou v daném roce, aby byly vyloučeny kontakty s neplatnými informacemi apod. Čištění kontaktů musí být naplánováno tak, aby měl supervizor dostatek času pro manuální potvrzení duplicit a následně mohlo dojít k vygenerování statistik s výsledky čištění pro Firmu B.

5. ***Manuální potvrzení duplicit***

Došlo-li během automatického čištění nových kontaktů k nálezu potenciálně duplicitních kontaktů, například na základě 80% shodných znaků jména (viz kapitola: 2.6 Čištění nových kontaktů), provede supervizor manuální potvrzení těchto duplicit. V případě, že k tomuto potvrzení nedojde, budou tyto kontakty považovány za neduplicitní a zařadí se do fronty k obvolání.

6. ***Výběr kontaktů pro obvolání***

Z vyčištěných kontaktů bude vybraná sada, která bude určena k telefonickému dotazování. Výběr se řídí souborem pravidel tak, aby byly pokud možno naplněny plány pro všechny partnery Firmy B. (Podrobnější informace o výběru kontaktů pro obvolání lze nalézt v bakalářské práci Filipa Šamánka).

7. ***Generování statistik z čištění kontaktů pro Firmu B***

Statistiky s výsledky čištění kontaktních informací budou předány v podobě XLS a CSV[11] reportů prostřednictvím webového rozhraní pracovníkům Firmy B.

[11] CSV – Comma Separated Values, textový formát dat oddělených čárkou

8. Kontrola plnění plánu

Pokud nejsou naplněny podmínky počtu rozhovorů (tj. nejsou naplněny plány pro jednotlivé partnery Firmy B), protože se zákazníkovi nebylo možné dovolat nebo se rozhovor nedokončil, bude výběr kontaktů pro aktuální dávku doplněn podle potřeby.

9. Vytočení hovoru

Každý agent má vlastní frontu kontaktů, kterou mu systém CSS automaticky doplňuje dle potřeby. Pro zahájení rozhovoru agent dle vlastního uvážení vybere z přidělené fronty jeden z kontaktů, přičemž dojde k navázání spojení se zákazníkem a agentovi bude zobrazen příslušný dotazník.

10. Nahrávání hovoru

Všechny hovory budou nahrávány a zálohovány. Pro operativní poslech budou v reálném čase dostupné hovory za posledních 6 měsíců. Celá historie hovorů bude uchovávána pro případ potřeby dohledání a poslech starších hovorů, které ale nemusí být přístupné v reálném čase.

11. Náslech rozhovoru pro klienta

Schválené hovory budou dostupné pro klienty Firmy B ve webovém rozhraní, kde si je budou moct poslechnout nebo stáhnout.

12. Vyplnění dotazníku

Agent call centra během telefonického rozhovoru vyplní dotazník. Dotazník může obsahovat otázky, které agent nebude vyplňovat (např. zda je dotazovaná osoba muž / žena). Tyto otázky vyplní až při následné kontrole dotazníku supervizor. V ideálním případě by měly být tyto otázky pro agenta skryté, aby se jimi nemusel při vyplňování dotazníku zabývat.

13. Kontrola dotazníku, editace, náslech

Supervizor call centra bude kontrolovat úplnost a gramatickou správnost vyplněných dotazníků a případně doplní chybějící údaje. V případě chyby nebo nejasnosti má možnost dotazník opravit, případně si poslechnout nahrávku rozhovoru a vyhodnotit ji.

14. Nastavení stavu hovoru

Na základě výsledku rozhovoru nastaví agent stav hovoru, aby bylo zřejmé, jestli byl hovor úspěšně dokončen, nebo je pouze rozpracován (např. kvůli přerušení spojení), případně jej označí speciálním stavem Robinson[9]. Stav hovoru může dodatečně upravit supervizor call centra, který bude vyplněný dotazník kontrolovat.

15. Generování statistik z dávky pro Firmu C

Pro Firmu C se na denní bázi budou exportovat data ve formátu SAV[12] (SPSS), která obsahují informace z vyplněných dotazníků. Přenášený soubor obsahuje vždy data od začátku kalendářního roku, aby bylo možné zaznamenat zpětné opravy dat.

16. Potvrzení oslovených kontaktů do Firmy B

Přehled kontaktovaných zákazníků bude předán zpět do Firmy B prostřednictvím *Služby pro výměnu dat*, čímž bude zajištěno, že jednou kontaktovaní zákazníci nebudou zařazeni do dalších probíhajících průzkumů.

2.4 Služba pro výměnu dat

Tato webová služba je určena k získávání a zpracování marketingových dat z databáze klientů Firmy B.

Aplikace CSS bude muset implementovat rozhraní pro komunikaci s touto službou. Z těchto důvodů zde bude představen základní princip komunikace a výměny dat.

2.4.1 Princip komunikace

Komunikace se *Službou pro výměnu dat* probíhá pomocí protokolu SOAP[13].

SOAP je protokol pro posílání zpráv XML[14] a je základem webových služeb. Ostatní standardy jako WSDL[15] a UDDI[16] vznikly až později po uvedení SOAPu a jen dále rozšiřují jeho možnosti a snadnost použití. SOAP umožňuje zaslání XML zprávy mezi dvěma aplikacemi a pracuje tedy na principu peer-to-peer. Zpráva je jednosměrný přenos informace od odesílatele k příjemci, ale díky kombinování několika zpráv můžeme pomocí SOAPu snadno implementovat běžné komunikační scénáře.

Nejčastěji se SOAP používá jako náhrada vzdáleného volání procedur (RPC), tedy v modelu požadavek / odpověď. Jedna aplikace pošle v XML zprávě požadavek druhé aplikaci, ta požadavek obslouží a výsledek zašle jako druhou zprávu zpět původnímu iniciátorovi komunikace. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká na požadavky klientů a v okamžiku, kdy přes HTTP přijde SOAPová zpráva, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán zpět klientovi jako odpověď. [1]

[12] SAV – přípona SPSS[3] formátu

[13] SOAP – Simple Object Access Protocol, protokol pro výměnu zpráv založený na XML přes síť

[14] XML – Extensible Markup Language, obecný značkovací jazyk, který byl vyvinut a standardizován komunitou W3C

[15] WSDL – Web Services Description Language, popisuje API webové služby, užíván zpravidla pro SOAP komunikaci

[16] UDDI – Universal Description, Discovery and Integration, adresářová služba pro podniky, sloužící k registraci a vyhledávání webových služeb

2.4.1.1 Autentizace

Služba pro výměnu dat vyžaduje autentizaci všech jejich konzumentů. Jednotliví konzumenti musí k webové službě přistupovat prostřednictvím svých aplikačních uživatelů. Identita aplikačních uživatelů je ověřována pomocí aplikačních certifikátů.

2.4.1.2 Autorizace

Aplikační uživatelé konzumentů mohou získat informace o studiích, pro které mají aktuálně přidělené oprávnění.

Příslušní aplikační uživatelé musí být přiřazeni k existující založené studii. Na základě tohoto přiřazení jsou vždy identifikovány studie, podle kterých se budou provádět autorizační ověření. Tímto způsobem je zabráněno, aby konzumenti nemohli neoprávněně získávat informace o studiích, se kterými nemají co do činění.

2.4.2 Definice rozhraní

Rozhraní webové služby obsahuje následující operace:

- *GetData* – na základě poskytnuté identifikace konzumenta a poskytnutých parametrů požadované studie vrátí informace pro danou studii. Po úspěšném přenosu informací dojde k zarezervování všech záznamů vrácených touto operací.
- *ConfirmData* – na základě poskytnuté identifikace konzumenta a sady záznamů s parametry identifikace zákazníka a dávky dojde k potvrzení dávky a uložení daného zákazníka do oslovených. Následně se odrezervují všechny záznamy ke každé dávce vyskytující se v sadě záznamů.

2.5 Ruční import kontaktů

Kromě standardního způsobu příjmu nových kontaktů přes *Službu pro výměnu dat* (viz kapitola: 2.4 Služba pro výměnu dat), musí aplikace CSS umožňovat příjem kontaktů také z jiných externích zdrojů. Například z důvodu odstávky *Služby pro výměnu dat*.

Pro tyto účely bude umožněno vždy před zahájením čištění kontaktů provést nahrání XLSX souboru s dodatečnými kontakty. Systém CSS následně tento soubor zpracuje a nové kontakty zahrne do zpracovávané dávky (do čištění kontaktů a následnému obvolávání).

Takto získané kontakty bude možno nahrávat vždy nejpozději hodinu před spuštěním automatického čištění kontaktů. Budou-li kontakty nahrány později, budou kontakty přiřazeny do následující nevyčištěné dávky.

2.6 Čištění nových kontaktů

Nově získané kontakty je potřeba před použitím zkontrolovat a očistit na základě sady pravidel uvedených níže v daném pořadí. Kontakty, které pravidla vyloučí z dalšího použití, zůstávají v databázi CSS z identifikačních důvodů a ponesou informaci s důvodem vyřazení.

Kontakty, které pravidly projdou, budou použity jako základní vzorek pro náhodný výběr pro telefonování.

Statistiky s důvody vyřazení kontaktů z telefonování se předávají zpět do Firmy B a jejím partnerům k revizi a jako podklad ke zlepšení kvality vstupních informací.

2.6.1 Kontrola telefonních čísel

Každý kontakt obsahuje pole pro telefon na zákazníka, mobilní telefon a pole se souhrnem telefonních čísel.

Telefonní čísla se přitom mohou v jednotlivých polích opakovat. Pole se souhrnem telefonních čísel obsahuje čísla použitá v ostatních polích, ale může obsahovat i další čísla navíc.

Před čištěním telefonních čísel je tedy potřeba vybrat všechna telefonní čísla z celého kontaktu, která jsou unikátní a ověření všech pravidel se provádí na takto vybraném setu.

Neobsahuje-li kontakt ani jedno vhodné telefonní číslo pro potřeby CSS, je nutné kontakt vyřadit z obvolávání.

2.6.1.1 Výčet pravidel pro telefonní čísla

1. Telefonní číslo u kontaktu je vyplněné a musí mít platný formát (v ČR 9 cifer bez mezinárodní předvolby nebo s předvolbou 00420 nebo +420). Jednotlivé čísla přitom mohou obsahovat i jiné znaky než čísla a znak "+", (například "/", "-" aj.) a stále se jedná o platné číslo.
2. V případě, že se jedná o číslo do zahraničí, lze toto číslo považováno za platné, ale nebude použito k obvolávání.
3. Telefonní číslo v rámci ČR nesmí být na placenou linku. Telefonní číslo bude považováno za placené, bude-li začínat cifrou 0, 1, 8 nebo 9.
4. Telefonní číslo nesmí být obsaženo v seznamu čísel, která nejsou považována za platná. Tento seznam budou udržovat pracovníci call centra.
5. Telefonní číslo nesmí nepatřit partnerovi Firmy B. Toto pravidlo lze sloučit s předchozím tak, že se telefonní čísla partnerů doplní do seznamu neplatných čísel.

2.6.2 Kontrola vyplněné osoby

Každý kontakt obsahuje pole se jménem a příjmením zákazníka, kontaktní osoby a případně dalších osob.

Pro potřeby CSS musí kontakt obsahovat alespoň jednu vyplněnou osobu, která bude použita k obvolávání, v opačném případě je nutné kontakt vyřadit z obvolávání.

2.6.3 Kontrola na partnera Firmy B

Zákazník určeným k obvolávání nesmí být partnerem Firmy B. Bude-li kontakt obsahovat vyplněné IČO[17], bude provedena kontrola, zdali toto IČO nepatří některému z partnerů Firmy B. V případě, že bude IČO nalezeno v seznamu partnerů, bude kontakt vyřazen z obvolávání.

Seznam partnerů Firmy B budou spravovat pracovníci call centra.

2.6.4 Kontrola na Robinsona[9]

Kontakt se nesmí nenacházet v databázi Robinsonů. Najde-li se shoda s některým z Robinsonů, je potřeba kontakt odstranit z obvolávání.

Porovnání se provádí pomocí všech telefonních čísel a jména / příjmení kontaktované osoby. Samotné porovnání pak probíhá v několika stupních:

1. Pokud je shoda v telefonním čísle, kontakt je shodný.
2. Pokud není shoda v telefonním čísle, ale je shoda celého jména (80% shodných znaků), posuzuje se shoda manuálně.
3. Pokud není shoda v telefonním čísle ani celém jméně, ale je shoda příjmení a adresy osoby (80% shodných znaků), posuzuje se shoda manuálně.

Databáze Robinsonů je samostatná evidence kontaktů, kde si zákazník nepřál být kontaktován a oslovován tímto průzkumem. Zařazení nového kontaktu do databáze provede agent call centra manuálně na základě průběhu telefonického rozhovoru se zákazníkem.

2.6.5 Kontrola duplicitních kontaktů

V rámci jedné dávky je nutné zachovat ve výběru pro volání pouze unikátní kontakty, tj. je nutné vyřadit duplicitní kontakty, které mohly vzniknout např. tím, že zákazník navštívil během jednoho týdne dva různé partnery Firmy B.

Kontrola se provádí na základě shody platných tel. čísel dvou různých kontaktů.

[17] IČO – Identifikační Číslo Obchodníka

2.7 Systém dotazníků

Systém dotazníků je nedílnou součástí každého call centra. Nezáleží přitom, zdali call centrum provádí výzkum trhu, technickou podporu, nebo se angažuje v úplně jiném odvětví. Vždy je totiž potřeba agentovi zobrazit formulář, do kterého bude postupně zaznamenávat reakce zákazníka. Takto nashromážděné informace jsou pak vyhodnoceny a případně předány k dalšímu zpracování.

V případě CSS bude dotazník sehrávat další důležitou roli, a sice že na základě již získaných reakcí dotazované osoby bude agentovi určovat, jaké další otázky má dotazované osobě pokládat.

2.7.1 Editace a tvorba nových dotazníků

Většina výstupních reportů CSS je přímo závislá na struktuře jednotlivých dotazníků. Dojde-li proto ke změně struktury dotazníků je potřeba rovněž příslušným způsobem upravit chování určitých reportů, na které má tato změna vliv.

Vzhledem k tomu, že takovouto úpravu reportů může zajistit pouze developer systému CSS, bude mít developer rovněž na starost editaci a vytváření nových dotazníků pro CSS.

Z těchto důvodu nebude potřeba GUI[18] pro editaci a správu dotazníků pro supervizory.

2.7.2 Práce s dotazníky

Systém dotazníků CSS se bude starat o manipulaci s daty dotazníků (vyplněné reakce zákazníka). Bude umožňovat jejich ukládání, načítání z databáze a automatickou validaci vyplněných dotazníků s kontrolou, zdali jsou všechna přijatá data v definovaném rozsahu a formátu.

Tato validace bude prováděna jak na straně serveru, tak na straně klienta (agenta) např. za pomocí JS[19]. V případě, že budou detekovány chybně vyplněné hodnoty, systém neumožní další zpracování dotazníku, dokud nebude provedena jejich korekce.

[18] GUI – Graphical User Interface

[19] JS – JavaScript, multiplatformní, objektově orientovaný skriptovací jazyk, jež se vykonává na straně klienta

2.7.3 Vykreslení dotazníků

Systém dotazníků se bude rovněž starat o jejich vykreslení. Všechny dotazníky budou OnePage[20], tedy na jediné stránce. Vzhledem k tomu, že jsou dotazníky velmi rozsáhlé, budou před začátkem rozhovoru téměř celé skryté. Při následném rozhovoru s dotazovanou osobou se pak budou agentovi postupně odkrývat další prvky dotazníku v závislosti na již zaznamenaných reakcích.

Toho lze navíc jednoduše využít například k informování agenta o tom, že opomenul vyplnit povinné pole, nebo vyplněné pole není v požadovaném rozsahu či formátu. V případě, že agent udělá v dotazníku chybu, nebudou mu zobrazeny další prvky dotazníku a agent tak bude mít možnost snadno dohledat a opravit svou chybu.

Dále musí dotazníky umožňovat vkládání údajů o agentovi a zákazníkovi do textu (např. pomocí speciálních znaků v textu). Důvodem je to, aby výsledný text, který bude následně agent číst dotazované osobě, mohl být formátován následujícím způsobem:

„*Dobrý den. Jmenuji se Petr Novák a volám Vám z call centra společnosti ETA za účelem výzkumu celkové spokojenosti s naším autorizovaným partnerem Elektro Dendis (Bohumín 1, Čáslavská 244).*“

Struktura a funkcionalita vykresleného dotazníku se navíc bude lišit v závislosti na roli, která bude k dotazníku přistupovat.

2.7.3.1 Agent

Provádí rozhovor se zákazníkem a jeho reakce zaznamenává do dotazníku. Z tohoto důvodu musí mít možnost editovat dotazník, nicméně některé prvky dotazníků mohou být pro agenta skryté, případně neaktivní. Jejich vyplnění provede supervizor při následné kontrole dotazníku.

2.7.3.2 Supervizor

Provádí korekci dotazníku a případně vyplňuje chybějící údaje. Má neomezený přístup k dotazníku a může editovat všechny jeho prvky (pole s daty).

2.7.3.3 Klient

Může prohlížet vyplněné dotazníky se všemi vyplněnými prvky, nicméně nemůže provádět jejich korekci a ukládat dotazník.

[20] OnePage – aplikace operující na jediné stránce, o překreslování obsahu (např. při přechodu na jinou stánku) se stará např. JS skript, který načítá data ze serveru, nebo schovává a zobrazuje data na stránce

2.7.4 Scénáře

Dotazník musí navíc podporovat plánování scénářů. To znamená, že se výčet otázek, na které se bude agent ptát dotazované osoby, bude lišit v závislosti na již zaznamenaných reakcí dotazované osoby. Důvodem je například to, že pokud dotazovaná osoba uvedla, že výsledný produkt nekoupila, je zbytečné a nesmyslné ptát se na otázky jak je s tímto produktem spokojena a zdali jej doporučí svým známým.

2.8 Generování reportů

Jak již bylo dříve řečeno, výsledné call centrum se bude zabývat výzkumem spokojenosti zákazníků s poskytovanými službami autorizovaných partnerů Firmy B.

Z těchto důvodu bude nejdůležitější funkcionalitou výsledného call centra generování různých reportů a statistik, na jejichž základě bude Firma B vylepšovat své dosavadní nabízené služby.

Všechny reporty bude plně automaticky generovat systém CSS. Čas vygenerování jednotlivých reportů, bude vhodně zvolen tak, aby byly všechny reporty dostupné nejpozději na začátku dne jejich dalšího zpracování (viz kapitola: 2.2 Plánování dávek).

2.8.1 Seznam jednotlivých reportů

Firma A neumožnila v rámci této bakalářské práce zveřejnit podrobný rozbor obsahu jednotlivých reportů, proto zde bude uveden pouze výčet reportů, které by mělo generovat každé call centrum obstarávající CATI rozhovory.

2.8.1.1 Statistiky z čištění dat

Po provedení čištění nově přijatých kontaktů a manuálním potvrzení potenciálních duplicit a robinsonů supervizorem (viz kapitola: 2.6 Čištění nových kontaktů) budou vygenerovány soubory se statistikou stavu kontrol, které budou po kontrole supervizorem vystaveny na webovém rozhraní CSS. Jednotlivé soubory si pak mohou přes webové rozhraní stáhnout zástupci Firmy B s příslušným oprávněním.

Tento report bude sloužit jako podklad k analýze kvality dat, jež shromažďují partneři Firmy B o svých zákaznících a bude generován ve formátu XLSX a CSV.

2.8.1.2 Statistiky pro Firmu C

Výstupem každého dotazníku bude SPSS report generovaný na denní bázi. Tento report v sobě ponese informace o všech vyplněných dotaznících od začátku kalendářního roku a bude sloužit jako podklad k podrobné analýze všech zákazníků a výzkumu jejich spokojenosti se službami partnerů Firmy B.

Jedná se tedy o nejdůležitější report, který bude systém CSS generovat. Tento report bude automaticky uploadován systémem CSS do Firmy C, která poté report dále zpracuje a vyhotoví pro Firmu B komplexní statistiky jednotlivých partnerů.

Fakt, že bude report obsahovat data z celého roku, bude potřeba zohlednit při výběru prostředků pro jeho generování (viz kapitola: 4.3 SPSS formát).

2.8.1.3 Statistiky provedených hovorů

Dalším výstupem call centra bude report s výčtem provedených hovorů a jejich délek. Tento report se bude generovat vždy jednou měsíčně ve formátu XLSX a bude mimo jiné sloužit i k vyúčtování za telekomunikační služby (myšleno mezi call centrem a Firmu B, pro kterou je výzkum prováděn, nikoliv mezi call centrem a poskytovatelem VoIP[21] telefonie).

2.8.1.4 Další reporty

Další reporty, které bude systém CSS generovat jsou již nadstavbou výsledného call centra a jejich obsah podléhá firemnímu tajemství, proto nebude v této práci popsán ani jejich význam pro Firmu B.

Dále je potřeba zohlednit fakt, že se v průběhu činnosti může skladba a obsah jednotlivých reportů na žádost Firmy B změnit. Výsledná aplikace by proto měla být navrhnuta tak, aby umožňovala dodatečné změny v reportovacím systému.

[21] VoIP – Voice over Internet Protocol

3. Nefunkční požadavky

Firma A rovněž předložila obsáhlý výčet nefunkční požadavků pro CSS.

Většina z těchto požadavků se týká HW a SW řešení systému CSS, způsob zálohování a způsobu zotavení při výpadku hlavního serveru. Plnění takovýchto požadavků bude zajišťovat externí firma, proto zde budou uvedeny pouze požadavky, které souvisí s rozsahem této práce.

3.1 Hardware

- K dispozici budou 2 servery (hlavní a záložní) a 20 stanic pro agenty a supervizory
- Minimální požadavky na server:
  - CPU: Intel/AMD, 4 jádra, frekvence 2.0 GHz, cache 4MB
  - RAM: DDR3, 8GB
  - 1x LAN
  - OS: Linux
- Minimální požadavky na stanici:
  - All-in-one PC
  - CPU: Intel/AMD, 2 jádra, frekvence 2.0 GHz, cache 2MB
  - RAM: DDR3, 2GB
  - 1x LAN
  - OS: Linux
  - Příslušenství (klávesnice, myš, sluchátka s mikrofonem)

3.2 Technologie

- CSS bude implementováno v jazyce PHP[22] a bude hostováno na vlastním serveru umístěném v prostorách call centra.
- Jednotlivé moduly např. pro filtrování kontaktů, generování reportů atd. mohou být implementovány i v jiných jazycích např. C/C++[23], Java[24] atd.
- Pro účely CSS bude použita databáze MySQL[25].
- Aplikace poběží na některé Linuxové distribuci.

[22] PHP – zkratka z PHP: Hypertext Pre-processor (původně z Personal Home Page Tools), jazyk pro tvorbu internetových prezentací

[23] C/C++ – nízko úrovňový, kompilovaný programovací jazyk pro operační systém Unix

[24] Java – objektově orientovaný programovací jazyk společnosti Oracle (dříve Sun Microsystem) nezávislý na platformě

[25] MySQL – databázový systém, vytvořený švédskou firmou MySQL AB, nyní vlastněný společností Sun Microsystems

3.3 Přístupy

- HTTPS[26] - přístup nebude limitovaný, certifikát poskytne Firma B.
- SFTP[27] - přístup bude limitován pouze na určité IP adresy, příslušené certifikáty dodá Firma A.
- SSH[28] - stejné jako u SFTP

3.4 Zálohování

- Jednou denně bude probíhat kompletní záloha aplikace (databáze a souborový systém).
- Pro případ výpadku serveru bude k dispozici záložní server, který bude dostupný do dvou hodin. Databáze z hlavního serveru bude online replikována na server záložní a souborový systém bude synchronizován jednou denně.

3.5 Další požadavky

- Aplikace umožní současnou práci až 20 agentů.
- Každý týden bude přijato až 11 000 kontaktů.
- Kontakty a záznamy o hovorech se budou archivovat minimálně dva roky.

[26] HTTPS – Hypertext Transfer Protocol Secure, zabezpečený protokol pro internetové prezentace
[27] SFTP – Secure File Transfer Protocol, zabezpečený protokol pro přenos souborů
[28] SSH – Secure Shell, zabezpečený protokol pro komunikaci v počítačových sítích

4. Analýza použitých technologií

Jak již bylo dříve uvedeno ve funkčních a nefunkčních požadavcích, výsledná aplikace pro systém CSS poběží na vybrané Linuxové distribuci s MySQL databázovým systémem a pro implementaci jádra systému CSS bude použit jazyk PHP. V této kapitole proto budou rozebírány technologie, které těmto požadavkům vyhovují.

Dále bude při výběru možných technologií kladen důraz na další rozvoj a možnosti CSS, jako například migraci na Windows server, či jiný databázový systém.

4.1 Práce s XLS soubory

Většina reportů CSS má být dle zadání ve formátu XLS / XLSX. PHP nativně nepodporuje tento datový formát aplikace Microsoft Office Excel. Pro práci s XLS soubory proto musí být využita některá z dostupných XLS knihoven.

Knihoven pro práci s XLSX, resp. s jeho starší verzí XLS je pro PHP celá řada, nicméně asi nejzajímavější a nejpoužívanější jsou knihovny Simple XLSX a PHPExcel, jejichž výhody a nevýhody budou dále rozebrány.

4.1.1 Simple XLSX

Simple XLSX je jednoduchá PHP knihovna určená ke čtení a zápisu XLSX souborů. [2]

4.1.1.1 Výhody

- Nespornou výhodou knihovny Simple XLSX je její velmi jednoduché API pro manipulaci s XLSX soubory.
- Velmi jednoduché a rychlé čtení a zápis do XLSX souborů.
- Knihovna je velmi malá a má malé nároky na prostředky.
- Knihovna je OpenSource[29].

4.1.1.2 Nevýhody

Z uvedených výhod zároveň plynou nevýhody této knihovny.

- Díky velmi jednoduchému API knihovna umožňuje pouze čtení a zápis do XLSX souborů a dokumenty nelze stylovat, určovat formát dat aj.
- Knihovna nepodporuje starý formát aplikace Microsoft Office Excel 97-2003 XLS.

[29] OpenSource – licence software, která umožňuje, při dodržení jistých podmínek, uživatelům zdrojový kód zdarma využívat

4.1.2 PHPExcel

PHP Excel je robustní PHP knihovna pro práci s XLS a XLSX soubory. Mimo jiné také podporuje práci s formáty CSV, Libre / OpenOffice Calc .ods, Gnumeric, PDF, HTML a mnoha dalšími. [3]

4.1.2.1 Výhody

- Podpora nejen formátů XLS a XLSX, ale i CSV, PDF a mnoha dalších.
- Podporuje stylování dokumentu, možné formáty dat v XLS / XLSX souboru a mnoho jeho dalších funkcí, jako jsou například grafy, obrázky aj.
- API knihovny je velmi dobře zdokumentované.
- Knihovna má rovněž rozsáhlou a aktivní komunitu na fóru.
- Stejně jako knihovna Simple XLSX je i knihovna PHPExcel OpenSource.

4.1.2.2 Nevýhody

- Oproti Simple XLSX podstatně složitější API.
- Podstatně vyšší nároky na prostředky.

4.2 PHP database layer

PHP nativně podporuje velký výčet různých databázových systémů. Mezi nejznámější z nich patří například PostgreSQL[30], SQLite[31], MS SQL[32], Oracle[33] a MySQL.

Problém s databázovými systémy je ovšem v tom, že mohou mít (a také mají) rozdílnou syntaxi jednotlivých dotazů. Například u MySQL databáze se počet vybraných záznamů z tabulky omezí pomocí příkazu LIMIT a OFFSET, kdežto u databáze Oracle aj. je potřeba použít příkaz ROWNUM.

Z těchto důvodů byla vytvořena celá řada různých PHP frameworků, které se snaží tento problém odstranit a umožnit jednotný zápis dotazů, které se pak v závislosti na zvolené databázi přeloží do jazyka, kterému daná databáze rozumí.

[30] PostgreSQL – multiplatformní objektově-relační databázový systém, šířen pod licencí MIT
[31] SQLite – multiplatformní relační databázový systém, šířen pod licencí public domain
[32] MS SQL – Microsoft SQL Server, relační databázový systém pro operační systém Microsoft Windows
[33] Oracle – multiplatformní databázový systém společnosti Oracle Corporation, nabízí pokročilé možnosti zpracování dat s vysokým výkonem

4.2.1 Doctrine

Doctrine je ORM[34] framework, který nabízí vysokou míru abstrakce databázové vrstvy za použití objektově orientovaného přístupu k datům, díky čemuž umožňuje pracovat s daty jako s objekty. [4]

Doctrine používá vlastní dotazovací jazyk zvaný DQL (Doctrine Query Language). Dotazy zapsané v tomto jazyce se před odesláním do databáze přeloží v závislosti na cílové databázi, takže odpadá potřeba úpravy SQL dotazů při migraci na jiný databázový systém.

4.2.1.1 Výhody

- Nízká náročnost na konfiguraci.
- Doctrine umí generovat objekty pro entity databáze z již existující databáze. Programátor pak pouze podle potřeby doimplementuje další funkcionalitu objektů.
- Databázová nezávislost.
- Defaultní použití transakcí u databází, které je podporují.
- Podpora stromových struktur dat.

4.2.1.2 Nevýhody

- Nevhodné pro složité dotazy, které jsou náročné na výkon databáze.
- Doctrine má velmi vysoké nároky na prostředky (viz test výkonu vystavený na stránce *SourceForge.net*, kde je porovnáván framework LightOrm s frameworky Doctrine a Propel [5]).
- Hodí se spíše pro menší až střední projekty, kde není kladen velký důraz na výkon.

4.2.2 Nette database connector

Jelikož výsledné řešení CSS bude postaveno na Nette Frameworku (viz kapitola: 5.2.1 Nette Framework), bylo by vhodné zmínit se i o databázovém layoutu, který nabízí. [6]

Tento layout stejně jako Doctrine využívá ORM techniku k manipulaci s daty v databázi. K popisu struktury databáze ale na rozdíl od Doctrine nepoužívá objekty zastupující entity databáze, ale využívá její reflexe. Tento přístup má ovšem záporný vliv na výkon frameworku, což se částečně snaží vyrovnat za pomoci cache.

Zároveň syntaxe aktuální verze neumožňuje nastavení složitějších vazeb mezi tabulkami databáze, kdy je potřeba referovat na cizí klíč z tabulky, který je tvořen více než jedním sloupce.

[34] ORM – Object-Relational Mapping, programovací technika, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem

4.2.2.1 Výhody

- ORM přístup.
- Integrováno přímo v Nette Frameworku.
- Velmi jednoduchá konfigurace a použití.
- Podpora MySQL, PostgreSQL, SQLite a MS SQL.
- Cachování dotazů.

4.2.2.2 Nevýhody

- Zhruba stejné nevýhody jako u Doctrine.
- Na rozdíl od Doctrine nepodporuje Oracle databázi a mnoho dalších.

4.2.3 Dibi

Dibi je velmi jednoduchá a nenáročná knihovna pro komunikaci s databází. I když se jedná o velmi malou knihovnu, podporuje všechny základní operace pro práci s daty a přidává k nim i několik dalších, jako je například vrácení pole, kde klíč a hodnotu tvoří vybrané sloupce z tabulky. [7]

Dále nabízí ochranu proti SQL Injection[35] a při využití fluent syntaxe umí generovat SQL kód pro požadovaný databázový systém.

Využití fluent syntaxe je navíc zajímavé i z toho důvodu, že při něm nezáleží na pořadí jednotlivých příkazů. Můžou se tak třeba nejprve definovat omezení na data (např. vybrání pouze prvních 10 záznamů) a až pak určit, z jaké tabulky budou data pocházet.

4.2.3.1 Výhody

- Malá nenáročná knihovna.
- Jednoduché a intuitivní API.
- Fluent syntaxe.
- Podpora MySQL, PostgreSQL, SQLite, MS SQL, Oracle a mnoha dalších.
- Široká škála podporovaných databázových systémů.
- Možnost tvorby optimalizovaných dotazů.

4.2.3.2 Nevýhody

- Nepodporuje ORM.

[35] SQL Injection – technika napadení databáze přes neošetřený vstup a vykonání vlastního kódu

4.3 SPSS formát

PHP nativně nepodporuje práci s datovým formátem SPSS, navíc pravděpodobně ani neexistuje žádná PHP knihovna, která by umožňovala práci s tímto formátem. Z těchto důvodů bylo potřeba najít jiný způsob jak splnit požadavky Firmy B.

4.3.1 SPSS Writer

SPSS Writer je knihovna napsaná v jazyce Java, kterou vytvořila firma pmStation. Jedná se o velmi jednoduchou knihovnu určenou pouze k zápisu SPSS souborů. Její obdobou je pak knihovna SPSS Reader, která slouží ke čtení SPSS souborů. [8]

Výhodou této knihovny je, že kromě zápisu SPSS souboru také umí vytvořit SPSS z načteného CSV, nebo XML souboru.

Další výhodou je pak, že je napsána v multiplatformním jazyce Java, pro kterou existují konektory pro MySQL a jiné databáze.

Obrovskou nevýhodou této knihovny je ale její pořizovací cena. Ta se totiž aktuálně pohybuje pro serverovou licenci kolem 400$ na jeden hardware. Vzhledem k tomu, že v CSS budou k dispozici celkem dva servery (hlavní a záložní), pohybovala by se pořizovací cena kolem 800$.

4.3.2 XML to SPSS Converter

Dalším možným řešením je XML to SPSS Converter od autora Jorana Jessuruna napsaný v jazyce C++ pro operační systém Windows. [9]

Tato velmi jednoduchá aplikace umožňuje převod XML souboru do datového souboru SPSS. Možným řešením problému by tedy mohlo být nejprve vygenerovat požadovaný XML soubor s daty v PHP a posléze spustit tuto aplikaci, která XML soubor převede do formátu SPSS.

Výhodou této aplikace je, že je OpenSource, čímž bohužel výčet jejich kladů končí.

Záporů lze ale u této aplikace najít podstatně více. Hned prvním může být fakt, že je tato aplikace určena pro operační systém Microsoft Windows, kdežto dle specifikací bude na serveru CSS nainstalována Linuxová distribuce. Výsledné řešení by se pak muselo kombinovat například s Wine[36].

Další nevýhodou tohoto řešení je, že tato aplikace nepodporuje všechny požadované datové typy SPSS souboru.

Třetím a největším záporem je pak to, že aplikace nepodporuje české kódování znaků.

[36] Wine – aplikace umožňující spouštění aplikací určených pro operační systém Microsoft Windows na systému Linux

4.3.3 Vlastní implementace SPSS Writeru

Dalším možným způsobem, jak plnit požadavky obdržené od Firmy B je implementace vlastního SPSS Writeru v jazyce C/C++.

Popis datové struktury SPSS souborů společnost IBM nezveřejnila, nicméně na své oficiální internetové stránce www.ibm.com nabízí po provedení registrace stažení C/C++ knihoven pro práci s SPSS formátem. Tyto knihovny lze dále šířit a bezplatně používat i pro komerční účely.

Popis API pro tento I/O modul lze rovněž nalézt na oficiálních stránkách společnosti IBM, nebo například v elektronické knize SPSS Input/Output Module, která je dostupná z tohoto zdroje [10].

4.4 Návrh a správa dotazníků

Systémů pro návrh a správu dotazníků pro PHP existuje velké množství. Většina z těchto řešení navíc nabízí GUI pro návrh a správu jednotlivých dotazníků, plánování scénářů či dokonce i generování různých statistik.

4.4.1 jForm

jForm je velmi jednoduchý a nenáročný OpenSource engine pro vytváření a generování dotazníků. Tento engine nemá vlastní GUI, a proto jednotlivé dotazníky musí vytvářet developer aplikace (což ovšem není v rozporu s funkčními požadavky). Pro popis elementů dotazníku využívá asociativních polí v PHP. Možným řešením by tedy mohlo být definovat strukturu dotazníku například v JSONu, který by pak příslušný PHP skript načetl a převedl na asociativní pole pro tento framework. [11]

4.4.1.1 Výhody

- Jednoduchá tvorba a správa dotazníků.
- Minimální nároky na prostředky.
- Aplikace filtrů na vstupní data a jejich validace.
- Možnost snadného rozšíření o další validační pravidla a dotazníkové entity.
- OpenSource.

4.4.1.2 Nevýhody

- Absence validace na straně klienta (agenta).
- Nepodporuje scénáře
- Neumí nativně zobrazovat/skrývat entity dotazníku v závislosti na roli uživatele.

4.4.2 Nette Forms

Jelikož výsledný systém CSS bude postaven na Nette Frameworku (viz kapitola: 5.2.1 Nette Framework), je vhodné představit prostředky pro generování dotazníků za pomoci Nette Forms, který ve výsledku nabízejí velmi podobnou funkcionalitu, jako výše představený jForm engine. [12]

Na rozdíl od jFrom nepoužívá pro popis struktury dotazníku asociované pole, ale definované objekty. Tím se sice přichází o možnost mít popsaný dotazník v externím JSON souboru, ale tento nedostatek lze vykompenzovat implementací pomocného rozhraní, které na základě JSON souboru vytvoří a nakonfiguruje výsledné objekty.

Velká výhoda Nette Forms ale spočívá v již integrované podpoře validace dotazníků pomocí JS skriptu na straně klienta (agenta).

4.4.2.1 Výhody

- Jednoduchá tvorba a správa dotazníků.
- Pokročilá validace a kontrola vstupních dat.
- Relativně nízké nároky na prostředky.
- Validace vyplněných dat i na straně klienta.
- Integrace v Nette Frameworku.

4.4.2.2 Nevýhody

- Nepodporuje scénáře.

4.4.3 Komerční aplikace

Dále existuje nepřeberné množství komerčních aplikací, které umožňují editaci a správu dotazníků, přičemž většina z nich je určena k provádění různých výzkumů a k tomuto účelu zároveň umožňuje generování různých reportů a statistik. Nejzajímavější z nich jsou pak pravděpodobně Formstack [13], JotForm [14] a MachForm [15]. Všechny tři aplikace si jsou vesměs rovnocenné, proto zde budou shrnuty jejich specifikace dohromady.

Všechny tři uvedené aplikace nabízí GUI pro jednoduché vytváření a správu dotazníků. Dále všechny podporují plánování scénářů, a v závislosti na roli uživatele umožňují skrytí, nebo deaktivaci jednotlivých elementů dotazníků.

Vkládání nových entit do dotazníků se provádí jejich přetáhnutím z palety dostupných prvků na místo, kde mají být vloženy. Stejným způsobem pak probíhá jejich přemisťování.

Jednotlivé vytvořené entity je navíc možno klonovat, nebo si předdefinovat různé seznamy čímž se tvorba/úprava formuláře podstatně zrychlí.

4.4.3.1 Výhody

- Přehledné a intuitivní GUI pro práci s dotazníky.
- Jednoduchá tvorba a editace dotazníků.
- Podpora scénářů.
- Podpora validace na straně klienta (agenta).

4.4.3.2 Nevýhody

- Takovéto aplikace bývají většinou hostované (vyjma MachForm) a uživatel tak nemá přímý přístup k datům, ale pouze k reportům a statistikám.
- Definování nových komponent dotazníků většinou není podporováno, nebo je velmi složité.

5. Návrh řešení

V této kapitole bude nejprve stručně popsán návrh hardwarového řešení a dále pak bude následovat výčet vybraných technologií pro systém CSS, jehož jádro bude tvořit Nette Framework.

Zdůvodnění, proč byl vybrán Nette Framework pro jádro systému CSS a analýzu konkurenčních frameworků lze najít v bakalářské práci Filipa Šamánka.

V této práci bude pouze uveden základní popis tohoto frameworku a jeho hlavní přednosti, které jej odlišují od ostatních.

V závěru této kapitoly pak bude navrhnut způsob řešení některých atypických funkcionalit výsledného CSS.

5.1 Vybraný hardware

Hardware obstará a nakonfiguruje dle potřeby a zadaných specifikací externí firma. Tato firma se bude rovněž starat o jeho správu a bude řešit technickou podporu. Zároveň bude mít na starost zálohování serveru a replikaci hlavního serveru na server záložní.

Z těchto důvodů zde bude pouze uvedena konfigurace výsledného HW tak, jak nám je zaslala tato firma.

5.1.1 Server

- HP ProLiant DL360 G4
- CPU: Intel® Xeon™ 3.0 GHz/800MHz, 2 MB L2
- Chipset: Intel chipset is E7520
- RAM: DDR3 1333 MHz, 12 GB
- OS: Linux Ubuntu Server
- HDD: 2x 750 GB Serial ATA
- LAN: 2x 100/1000 Mbps

5.1.2 Stanice pro agenty

- Lenovo ThinkCentre S710
- CPU: Intel Core i3 3.4 GHz
- Chipset: Intel H61
- RAM: DDR3 1600 MHz, 4 GB
- OS: Linux Mint
- HDD: 500 GB Serial ATA

5.2 Použité technologie

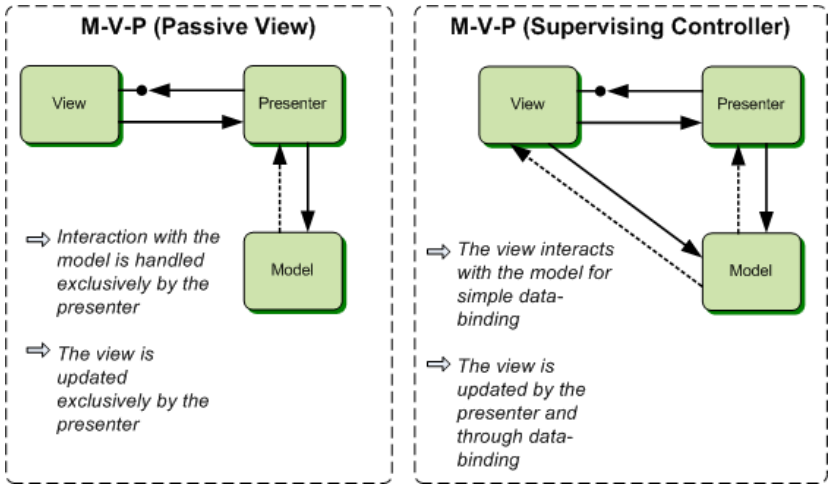
Výběr technologií probíhal na základě funkčních a nefunkčních požadavků, které zaslala Firma B a z analýzy použitých technologií (viz kapitola: 4 Analýza použitých technologií).

Dále byla při výběru zvolených řešení zohledněna vlastní zkušenost s jednotlivými technologiemi, jejich využití v rámci tohoto projektu a jejich licenční politika.

5.2.1 Nette Framework

Nette Framework je populární nástroj pro vytváření webových aplikací v jazyce PHP 5, který původně pochází od českého autora Davida Grudla. [16]

Nette Framework je postaven na architektuře MVP[37], která usnadňuje rozčlenění aplikace na její jednotlivé vrstvy. Vztah jednotlivých vrstev této architektury lépe demonstruje přiložený Obrázek 2.



Obrázek 2 - Architektura MVP

Nette Framework navíc disponuje skvělým ladícím nástrojem pro odchytávání a zpracování chyb. Tento ladící nástroj mimo jiné umí logovat chyby aplikace a v případě detekce chyby zaslat developerovi notifikaci v podobě emailové zprávy. Dále umožňuje přehledné vypisování obsahu proměnných a ve spolupráci s Nette Database, nebo Dibi zobrazovat zaslané dotazy na databázi s informacemi o době transakce a množství přijatých dat.

Z nezávislého testu, který byl zveřejněn na portálu *Root.cz* navíc vyplývá, že je Nette Framework jedním z nejrychlejších frameworků, které jsou aktuálně běžně využívány. [17]

[37] MVP – Model View Presenter

5.2.2 JavaScript

JavaScript (zkráceně JS) je multiplatformní, objektově orientovaný skriptovací jazyk, který se využívá jako interpretovaný programovací jazyk pro tvorbu WWW stránky.

Důvodem pro výběr JavaScriptu jakožto client-side bylo, že jej nativně využívá Nette Framework a zakoupená šablona Core Admin [18], postavená na Bootstrap frameworku [19].

Dalším důvodem k výběru byly technologie, které jej využívají a které budou použity ve výsledné aplikaci. Konkrétně půjde o technologie *AJAX*, *jQuery* a *DataTables.*

V aplikaci CSS bude JavaScript využíván například k validaci dotazníků na straně klienta, nebo k zobrazování notifikací stránky bez nutnosti znovu-načtení (reloadu) stránky.

5.2.2.1 AJAX

AJAX[38] je využíván k vývoji interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich kompletního znovu-načítení (reloadu).

Dále jej lze využít ke komunikaci se serverem na pozadí stránky, čehož lze využít například pro komunikaci s ústřednou (vytáčení a ukončování hovorů), nebo k získávání dat pro knihovnu DataTable (viz kapitola: 5.2.2.3 DataTable).

5.2.2.2 jQuery

jQuery je JS knihovna usnadňující manipulaci s HTML prvky a řešící problémy s kompatibilitou JavaScriptu napříč všemi nejpoužívanějšími prohlížeči.

Navíc disponuje nástroji pro snadnou animaci HTML objektů, AJAXovou komunikaci a manipulaci s přijatými daty, či zachytávání událostí (například klik na prvek stránky, nebo pohyb kurzoru myši po stránce).

5.2.2.3 DataTables

DataTables je plug-in pro knihovnu jQuery. Jedná se o flexibilní nástroj umožňující jednoduchou tvorbu datových tabulek v kombinaci s HTML. [20]

Nad těmito tabulkami pak lze provádět operace jako je vyhledávání, řazení dle sloupců, či v případě, že tabulka obsahuje velké množství záznamů, jejich rozdělení na jednotlivé podstránky.

V kombinaci s AJAXem, lze pak data pro takto vytvořené tabulky načítat na pozadí stránky přímo ze serveru. To je obzvláště výhodné pro situace, kdy výsledná tabulka obsahuje tisíce až milióny záznamů. Práce JavaScriptu se tak přenese na server, nebo databázi, která je pro práci s takto velkým počtem záznamů podstatně lépe optimalizovaná.

[38] AJAX - Asynchronous JavaScript and XML

5.2.3 Dibi

Pro komunikaci s databází byla zvolena knihovna Dibi.

Hlavním důvodem pro její zvolení byla její vysoká rychlost a nízká náročnost na prostředky. Navíc s její pomocí lze snadno skládat optimalizované dotazy na databázi, což bude hrát velmi důležitou roli převážně při generování reportů a statistik pro CSS.

Dalším důvodem pak bylo, že kromě MySQL podporuje širokou škálu jiných databázových systémů (viz kapitola: 4.2.3 Dibi), díky čemuž by se v případě potřeby migrace na jiný databázový systém ušetřilo mnoho práce s přepisem SQL kódu.

Navíc lze pro Dibi knihovnu snadno implementovat další ovladače pro databáze, které tato knihovna nativně nepodporuje.

5.2.4 PHPExcel

Pro práci s Excel soubory byl vybrán framework PHPExcel. Výsledné řešení počítá se čtením a zápisem do XLSX i XLS souborů. Z těchto důvodů nemohl být použit framework Simple XLSX (viz kapitola: 4.1 Práce s XLS soubory).

5.3 Uživatelské role a systém oprávnění

Výsledná implementace systému CSS bude mít řadu uživatelských rolí.

Kromě rolí *Agent*, *Supervizor* a *Klient* bude systém navíc obsahovat role *Everybody*, *Cron*, *Guest*, *Member*, *VoIP* a *Support*. Důvodem je lepší rozčlenění jednotlivých uživatelů a zjednodušení správy jejich oprávnění. Schéma rozdělení rolí popisuje Obrázek 3.

Nette Framework nativně obsahuje základní systém pro autentifikaci a autorizaci uživatelů, proto bude tento systém využit a rozšířen dle potřeby o další funkcionalitu. [21]

5.3.1 Kontrola oprávnění

Kontrola uživatelových oprávnění bude prováděna v *BasePresenter*, od kterého budou dědit všechny ostatní presentery.

Při přístupu uživatele na stránku *BasePresenter* automaticky zkontroluje, má-li uživatel dostatečná oprávnění přistoupit k požadovanému presenteru a vykonat požadovanou akci.

Díky tomuto přístupu k oprávněním bude zaručeno, že uživatel nevykoná žádnou akci, ke které nemá přístup, nebo nepřistoupí k obsahu, který je mu odepřen.

Zároveň se také minimalizuje potřeba neustále kontrolovat uživatelovy oprávnění napříč celou aplikací CSS.

5.3.1.1 Způsob kontroly oprávnění

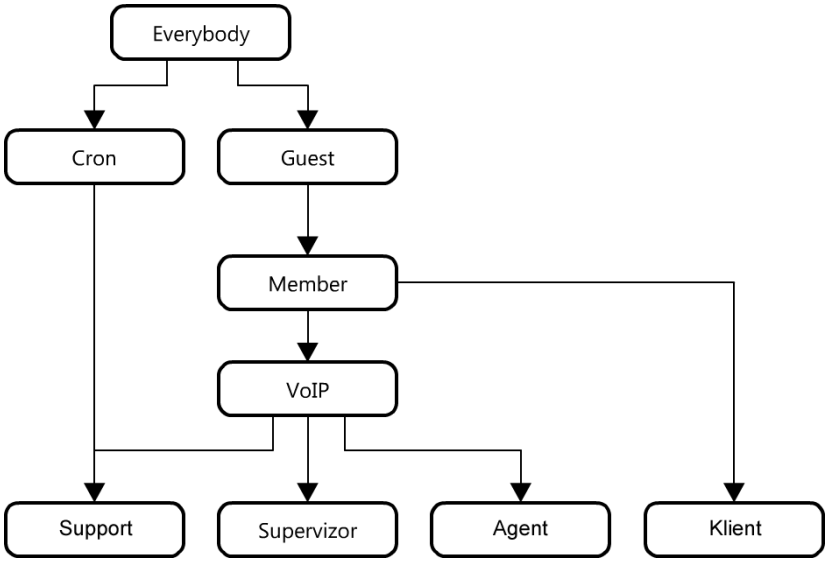
1. V případě, že uživatel nebude mít oprávnění vykonat požadovanou akci, ale bude mít oprávnění přistoupit k presenteru, bude automaticky přesměrován na jeho defaultní akci, která mu bude vždy povolena.

2. V případě, že nebude mít oprávnění přístupu na požadovaný presenter, ale bude přihlášen, bude přesměrován na *HomePagePresenter*, který je dostupný pro všechny přihlášené uživatele.

3. Nebude-li uživatel přihlášen, bude automaticky přesměrován na *SignPresenter* a vyzván k přihlášení do systému.

5.3.2 Role

Podrobným popisem pravomocí uživatelů se zabývala kapitola: 2.1 Uživatelské role. V této kapitole proto bude pouze zdůvodněno přidání nových rolí do systému a jejich schéma dědění (viz Obrázek 3).



Obrázek 3 - Schéma a způsob dědění uživatelských rolí

5.3.2.1 Everybody

Tato role bude tvořit základ všech ostatních rolí. Bude oprávněna přistupovat pouze k *ErrorPresenteru* a *SignPresenteru* a vykonávat na nich všechny akce.

5.3.2.2 Guest

Role *Guest* bude dědit od role *Everybody*. Každý nepřihlášený uživatel bude automaticky v této roli.

5.3.2.3 Cron

Pro účely automatického generování reportů, stahování kontaktů a dalších procesů byl vytvořen *CronPresenter*. Aby bylo možno snadno zamezit přístupu k tomuto presenteru, byla definována role *Cron* dědící od role *EveryBody*.

Při přístupu uživatele na tento presenter se provede kontrola, zdali je uživatel v roli *Cron*, nebo od ní alespoň dědí. V případě, že tato podmínka nebude splněna, bude uživatel automaticky přesměrován systémem na chybovou stránku 403 Acces Denied.

Kontrolu, zda je uživatel Cron bude provádět *BasePresenter*. V případě detekce uživatele v roli *Guest* provede kontrolu, zda uživatel přistupuje z localhostu a pokud dojde ke shodě, změní roli uživatele na roli *Cron*.

5.3.2.4 Member

Role *Member* bude dědit od role *Guest*. Z této role pak budou dědit všechny role přihlášených uživatelů. Navíc bude roli *Member* umožněn neomezený přístup na *HomePagePresenter*.

5.3.2.5 VoIP

Tato role bude sloužit pouze k tomu, aby systém rozpoznal, zdali má uživatel schopnost připojit se a komunikovat s ústřednou. Proto musí role *Agent*, *Supervizor* a *Support* od této role dědit.

5.3.2.6 Support

Role *Support* bude sloužit k přístupu servisního technika. Tato role bude mít povolen přístup na všechny presentery, včetně *CronPresenteru* (pro schopnost manuálně invokovat automatizované procesy) a možnost připojit se k ústředně. Z těchto důvodů dědí role *Support* od rolí *Cron* a *VoIP*.

Navíc bude mít role *Support* automaticky zapnutý debug-mod pro rychlejší a snadnější ladění aplikace CSS.

5.3.3 Autentifikace uživatelů

Uživatele *Agent*, *Supervizor* a *Support* je při přihlášení do systému CSS zároveň potřeba registrovat v ústředně, jinak by nebylo možné s ústřednou komunikovat. Z těchto důvodu bude model obstarávající autentifikaci uživatelů rozšířen o prostředky umožňující tuto funkcionalitu.

5.3.4 Autorizace uživatelů

Systém autorizace uživatelů byl upraven tak, aby spolupracoval s modulem *PermissionLoader*. Tento modul zajišťuje dynamické načítání oprávnění jednotlivých rolí a uživatelů z databáze.

V případě, že je systému oprávnění položen dotaz, zdali má uživatel potřebná oprávnění k vykonání určité akce, provede tento systém nejprve kontrolu, zda má nezbytné informace pro zpracování dotazu.

Pokud systém detekuje, že o daném uživateli tyto informace nemá, požádá o ně modul *PermissionLoader*. Ten pak podle potřeby načte potřebné informace z databáze a předá je zpět systému pro autorizaci.

5.4 Stahování kontaktů a jejich potvrzování do Firmy B

Stahování nových kontaktů a potvrzování oslovených kontaktů bude prováděno pomocí skriptu v *CronPresenteru*, který bude automaticky spouštět v daný čas Cron[39].

Pro získání kontaktů, či jejich potvrzení bude implementováno API, které zajistí komunikaci se *Službou pro výměnu dat* (viz kapitola: 2.4 Služba pro výměnu dat).

5.5 Filtrování kontaktů

Pro filtrování kontaktů byl zvolen jazyk Java. Hlavním důvodem je možnost paralelního zpracování čištění kontaktů. Díky tomu se podstatně zkrátí čas nezbytný k provedení této operace, což dokazuje i test v kapitole 7 Testování aplikace.

Filtrování kontaktů bude probíhat ve večerních hodinách, kdy provoz CSS omezen a nebude tak narušen plynulý chod CSS v případě zvýšených nároků na prostředky.

O automatické spouštění filtrování kontaktů se bude starat Cron, který v určený čas spustí skript v *CronPresenteru*. Následně dojde ke shromáždění informací o kontaktech, které je potřeba přefiltrovat a pomocí příkazu *execute* bude spuštěna Java aplikace, která provede výsledné čištění dat.

Po ukončení čištění dat script v *CronPresenteru* zkontroluje výstup Java aplikace a v případě detekovaných problémů zašle notifikační email developerovi aplikace.

Pokud při filtrování nedojde k žádnému problému, bude po ukončení vygenerován report s výsledky čištění kontaktů a zveřejněn na webovém rozhraní aplikace CSS.

[39] Cron - je softwarový démon, který v operačních systémech automatizovaně spouští v určitý čas nějaký proces

5.6 Generování reportů

Generování reportů bude probíhat stejným způsobem, jako filtrování kontaktů s tím rozdílem, že nebude potřeba spouštět externí aplikace (až na generování SPSS reportu, viz následující kapitola: 5.7 Generování SPSS reportů).

Dále bude systém zajišťující generování reportů umožňovat jejich zpětné přegenerování. V průběhu času totiž může dojít ke změně obsahu některého z reportů a může být proto potřeba některé reporty zpětně přegenerovat.

5.7 Generování SPSS reportů

Jelikož jsou dostupné prostředky pro generování SPSS reportů nevyhovující, bude potřeba implementovat vlastní aplikaci pro manipulaci s těmito datovými soubory.

Výsledné řešení bude implementováno v jazyce C++ za pomoci SPSS I/O knihovny, kterou poskytuje volně ke stažení IBM (viz kapitola: 4.3.3 Vlastní implementace SPSS Writeru).

Impulzem pro zahájení generování tohoto reportu bude spuštění patřičného skriptu v *CronPresenter*, který spustí výslednou aplikaci.

Po vygenerování reportu bude stejně jako u čištění kontaktů zkontrolována návratová hodnota aplikace a v případě detekce problémů zaslán notifikační email.

Pokud generování reportu proběhne v pořádku, dojde k jeho uploadu do Firmy C přes FTP a skript se ukončí.

5.8 Import kontaktů ze XLSX souborů

Po nahrání souboru s kontakty bude soubor uložen do *temp* adresáře. Tento uložený soubor pak bude přečten a jednotlivé kontakty z něj extrahovány a načteny do asociativního pole. Toto pole pak bude převedeno do formátu JSON a uloženo v *temp* adresáři.

Při extrahování kontaktů bude zároveň prováděna kontrola, zda kontakt obsahuje všechny nezbytné informace (jméno, telefon, datum návštěvy partnera Firmy B aj.) a zda jsou tyto data v požadovaném formátu a tvaru.

Následně bude vykreslena uživateli přehledná tabulka s načtenými daty a případně vyznačenými problémovými sekcemi (viz přiložený Obrázek 4).

Potenciální nepřesnosti v datech, které nemají vliv na správný chod CSS budou vyznačeny žlutou barvou a kritické chyby, které je nutno opravit barvou červenou.

Navíc při najetí kurzorem myši na vyznačený text bude zobrazena legenda s popisem notifikace, nebo chybová hláška.

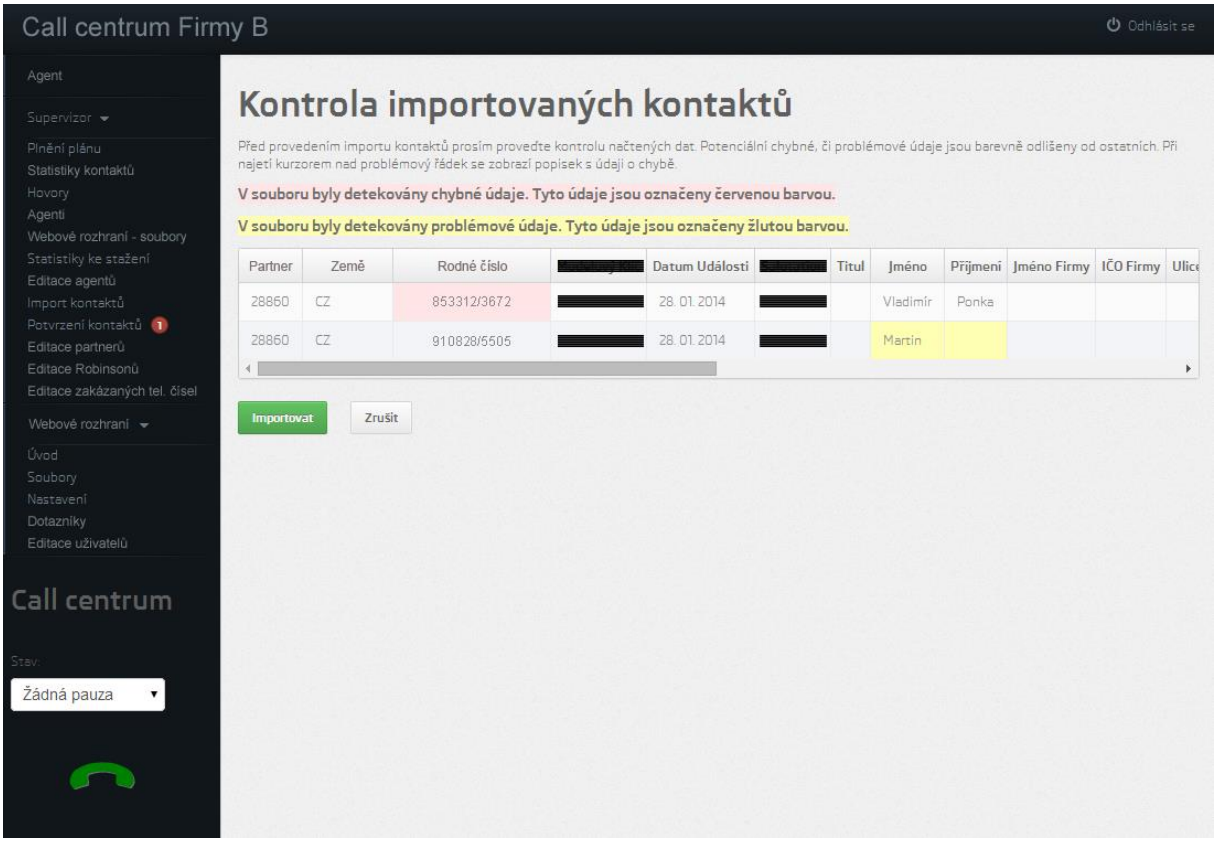
V případě detekce fatální chyby v datech (například pokud nebude rodné číslo, viz Obrázek 4) nebude umožněn import, dokud nebudou všechny chyby opraveny.

Po vizuální kontrole načtených dat ze souboru bude mít uživatel možnost buď import přerušit, nebo přejít na další krok.

Pokud se rozhodne proces přerušit, budou všechny příslušné dočasné soubory smazány a import kontaktů bude ukončen bez jakýchkoliv změn v databázi.

Pokud se uživatel rozhodne v procesu pokračovat, budou načteny již zpracované data z JSON souboru a dojde k jejich importu do systému CSS a uživatel bude informován o výsledku zpracování přehlednou hláškou.

Jelikož lze kontakty importovat pouze do dávky, která ještě nebyla přefiltrována a předána k dalšímu zpracování, bude uživatel před importem navíc informován, do které dávky budou kontakty přiřazeny a která dávka se aktuálně zpracovává.



Obrázek 4 - Ilustrace možného výpisu načtených dat ze souboru s kontakty

5.9 Návrh databáze

Databáze byla navržena tak, aby byla optimalizována pro rychlý chod systému CSS. Z těchto důvodů je občas porušena třetí norma a jednotlivé tabulky tak mohou obsahovat duplicitní data (viz Příloha 1, popisující strukturu databáze).

Dále zde budou popsány pouze tabulky, které se nějak týkají této práce. Ostatní tabulky má popsané ve své bakalářské práci Filip Šamánek.

5.9.1 Tabulka contact_new

Kontakty obdržené přes službu *Služba pro výměnu dat*, nebo ručně importované se ukládají do tabulky *contact_new*. Tato tabulka obsahuje surová data s příznaky o kvalitě dat, které jsou nastaveny během čištění kontaktů.

5.9.2 Tabulka contact

Po čištění kontaktů se do tabulky *contact* vloží ty kontakty, které byly určeny k dalšímu zpracování. Do této tabulky jsou zahrnuty pouze ty informace o kontaktu, které jsou nezbytné pro jeho další zpracování.

5.9.3 Tabulka owner

Tato tabulka obsahuje informace o partnerech Firmy B, pro které je prováděn výzkum spokojenosti zákazníků.

5.9.4 Tabulka disabled_phones

V této tabulce je umístěn seznam zakázaných telefonních čísel, která nesmějí být použity pro účely CSS (viz kapitola: 2.6.1 Kontrola telefonních čísel).

5.9.5 Tabulka robinson

V této tabulce je umístěn seznam robinsonů. Tato tabulka je rovněž určena výhradně pro filtrování kontaktů (viz kapitola: 2.6.4 Kontrola na Robinsona[9]).

5.9.6 Tabulka calls

Tato tabulka obsahuje výčet provedených hovorů. Dále nese informace o tom, zda byl vyplněn dotazník, jaký agent hovor prováděl a s jakým stavem rozhovor skončil.

5.9.7 Tabulka user

V této tabulce jsou informace o uživatelích CSS. Jejich jméno, autentifikační údaje a role uživatele, která má význam při autorizaci uživatele (viz kapitola: 5.3.4 Autorizace uživatelů).

5.9.8 Tabulka user_permission

V této tabulce jsou informace o dodatečných oprávněních jednotlivých uživatelů, díky čemuž lze dodatečně povolit, nebo zakázat různé akce konkrétním uživatelům nehledě na roli uživatele.

5.9.9 Tabulka role_permission

Tabulka je používaná pro nastavení oprávnění pro jednotlivé role uživatelů v systému.

5.9.10 Tabulka wrap_filled

Tabulka využívaná systémem dotazníků (viz kapitola: 2.7 Systém dotazníků). Nachází se v ní informace o tom, v jaké tabulce jsou uloženy data z dotazníků a některé informace z těchto tabulek, které jsou využívány při aplikaci filtru ve výpisu provedených hovorů (tuto funkcionalitu má popsanou ve své bakalářské práci Filip Šamánek).

5.9.11 Tabulky wrap_answers_*

Tabulky využívané systémem dotazníků (viz kapitola: 2.7 Systém dotazníků). Obsahují uložená data z jednotlivých dotazníků.

5.10 Návrh dotazníků

Pro vytváření a editaci dotazníků byl vybrán framework Nette Forms, který je součástí Nette Frameworku (viz kapitola: 4.4 Návrh a správa dotazníků), na jehož základě byly navrhnuty komponenty pro výsledné dotazníky.

Vzhledem k tomu, že nebylo požadováno GUI pro tvorbu dotazníků, je struktura výsledného dotazníku popsána v JSON souboru, který lze snadno editovat ve vhodném textovém editoru.

Formát JSON byl zvolen také z toho důvodu, že PHP umí pracovat s JSON souborem daleko rychleji, než například s formátem XML.

5.10.1 Model Wrap

Tento model na základě poskytnutých informací o požadovaném dotazníku, roli uživatele a případně ID uloženého dotazníku vrátí požadovaný nakonfigurovaný dotazník, který je určen k vykreslení na stránce.

Dále obsahuje funkci pro uložení vyplněného dotazníku do databáze systému CSS.

5.10.2 Vytváření a editace dotazníků

Dotazníky jsou definovány pomocí JSON souboru. Tento soubor nese informace o struktuře celého dotazníku a také podpůrný JavaScript, který se stará o základní validaci dotazníku a plánování scénářů.

Tento JSON soubor je pak pomocí modelu *WrapBuilder* načten a na základě struktury dotazníku, která je v něm popsána se sestaví celý dotazník, který serializuje a uloží do *temp* adresáře (viz kapitola: 5.10.3 Model WrapBuilder).

5.10.2.1 Definice dotazníku

Každý dotazník musí mít jedinečný název, podle kterého bude identifikován a načten.

Dále musí být definován seznam možných stavů dotazníku a nepovinně uvítací text.

Dále každý dotazník obsahuje výčet elementů a jejich nastavení, které budou použity pro sestavení struktury dotazníku.

5.10.2.2 Definice elementu

Každý element v dotazníku musí mít jedinečný název (ID), dále musí být specifikován typ elementu (např. text, label, timepicker aj.) a volitelně JavaScript kód pro obsluhu elementu či plánování scénáře.

Dalším volitelným atributem je pak pomocný text, který se zobrazí zároveň s elementem.

Další atributy jsou pak závislé na typu daného elementu, například u textového pole to může být atribut *maxlen*, který určuje maximální počet znaků, nebo u *selectboxu*, či *radioboxu* atribut *options*, který definuje možné odpovědi.

5.10.3 Model WrapBuilder

Jak již bylo dříve řečeno, *WrapBuilder* slouží k sestavení dotazníku podle jeho popisu v JSON souboru.

Před vyžádáním dotazníku nejdříve *WrapBuilder* provede kontrolu, zdali dotazník již nebyl jednou vytvořen a serializován. Pokud dotazník již byl serializován, provede kontrolu, zda nebyl aktualizován JSON soubor popisující strukturu požadovaného dotazníku (kontrola se provádí na základě data poslední aktualizace souboru). V případě že nebyl, načte serializovaný dotazník a vrátí jej. V případě že serializovaný dotazník není k dispozici, nebo byl aktualizován JSON soubor popisující strukturu dotazníku, načte tento JSON soubor a na základě jeho obsahu vytvoří nový dotazník který serializuje a vrátí.

Při zpracování JSON souboru se navíc provádí kontrola, zdali v něm jsou definovány všechny povinné atributy. Pokud dojde při vytváření dotazníku k chybě, například z důvodu chybějícího povinného atributu, nebo duplicitním názvu ID komponent, dojde k vyhození výjimky s podrobným popisem chyby.

Během vytváření dotazníku se zároveň shromažďuje veškerý JavaScriptový kód a informace o vstupech (prvkách formuláře) jednotlivých komponent dotazníku.

JavaScript kód je následně zkompilován (odstraněny všechny přebytečné znaky) a uložen do veřejného adresáře *www/javascripts/wrap*.

Dále je na základě nashromážděných informací o vstupech jednotlivých komponent vygenerován vzorový MySQL kód, který popisuje vzorovou tabulku pro data z dotazníku.

Tento MySQL kód je pak uložen do *temp* adresáře a může sloužit jako podklad pro tabulku dotazníku v databázi.

5.10.4 Komponenta WrapControl

Tato komponenta tvoří prototyp výsledného dotazníku a disponuje následujícími prostředky:

1. Možnost nahrazování speciálních symbolu v textu za data o zákazníkovi.
2. Nastavení módu pro vykreslení (liší se podle typu uživatele, viz kapitola: 2.7 Systém dotazníků).
3. Interface pro vytvoření dotazníku z načteného JSON souboru (viz kapitola: 5.10.3 Model WrapBuilder).
4. Nástroje pro serializaci dotazníků (viz kapitola: 5.10.3 Model WrapBuilder).
5. Interface pro nastavení odpovědí dotazníků (používáno při vykreslení uloženého dotazníku).
6. Interface pro získání zaznamenaných a validních reakcí zákazníka.
7. Renderování dotazníku.
8. Získání informací o vstupech dotazníku (používáno pro generování MySQL kódu pro tabulku, viz kapitola: 5.10.3 Model WrapBuilder).

5.10.5 Komponenta Wrap

Tato komponenta rozšiřuje komponentu *WrapControl* a výsledný dotazník doplňuje o vstupy pro zaznamenání stavu dotazníku, nebo času pro přeplánování.

Dále je použita pro stylování komponent dotazníku a nese část podpůrného JavaScriptového kódu, který například kontroluje před odesláním dotazník, zdali jsou vyplněny všechny požadované vstupy.

5.10.6 Komponenta Element

Tato komponenta je prototypem komponenty pro *WrapControl*, a musí od ní dědit všechny implementované komponenty dotazníku.

Disponuje následujícím API.

1. Nastavení rodiče (v tomto případě to bude *Wrap*).
2. Interface pro vytvoření komponenty z načteného JSON souboru (viz kapitola: 5.10.3 Model WrapBuilder).
3. Nástroje pro serializaci komponenty (viz kapitola: 5.10.3 Model WrapBuilder).
4. Možnost nahrazování speciálních symbolu v textu za data o zákazníkovi.
5. Renderování komponenty.
6. Získání informací o vstupech komponenty (používáno pro generování SQL kódu pro tabulku, viz kapitola: 5.10.3 Model WrapBuilder).

6. Implementace

Výsledná implementace aplikace systému CSS přesně odpovídá uvedenému návrhu řešení v kapitole 5 Návrh řešení, proto se tato kapitola zaměří na další funkcionalitu, která byla implementována nad rámec této práce.

6.1 Systém pro správu souborů

V rámci této práce byl rovněž implementován jednoduchý model zvaný *FileManager*, určený pro manipulaci se soubory systému CSS. Tento model se hojně využívá například při generování jednotlivých reportů, nebo k zabezpečenému stahování souborů obsahující citlivá data.

Tento model nabízí celkem tyto prostředky:

1. Automatické promazávání *temp* adresáře.
2. Práci s dočasnými soubory.
3. Výpis obsahu adresáře.
4. Zabezpečené stahování citlivých souborů.
5. Mazání souborů.
6. Nahrávání souborů.
7. Možnost vrácení absolutní cesty souborů či předdefinovaných složek.

6.2 Translátor

Při implementaci systému pro CSS bylo rovněž myšleno na jeho další využití a případné nasazení v cizině.

Z tohoto důvodu byl rovněž implementován translátor pro překlad textů a všechny texty, které se v aplikaci vykytují, jsou psány v anglickém jazyce. Z anglického jazyka se pak za pomocí translátoru překládají do požadovaného jazyka, takže při absenci překladu se zobrazí text alespoň v angličtině.

Tento translátor byl rovněž implementován v jazyce JavaScript a umožňuje tak překlad textů i na straně klienta.

6.2.1 Balíčková služba i18n

Pro účely překladu na serveru byla implementována balíčková služba *i18n*.

Tato služba se stará o lokalizaci aplikace, obsluhuje jazykové balíčky a vytváří instance translátorů pro komponenty a presentery aplikace.

Jednotlivé jazykové balíčky se pak nacházejí v těchto adresářích:

- app/i18n/<language>/components/*
- app/i18n/<language>/presenters/*
- app/i18n/extra/*

6.2.2 Implementované funkce

- Funkce pro změnu jazyka.
- Funkce pro zjištění aktuálně používaného jazyka a defaultního jazyka.
- Funkce pro vytvoření instance translátoru.
- Funkce pro obsluhu JavaScriptového translátor.

6.2.3 Pořadí načítání jazykových balíčků

Balíčky se načítají v závislosti na typu obdrženého zdroje. Je-li zdrojem instance presenteru, načtou se jazykové balíčky pro daný presenter a balíčky presenterů, od kterých tento presenter dědí.

Je-li zdrojem instance komponenty, načtou se obdobným způsobem jazykové balíčky pro danou komponentu. Má-li navíc komponenta nastavený rodičovský presenter, provede se rovněž načtení jazykových balíčků pro tento presenter.

Je-li povoleno použít v případě chybějícího překladu překlad z defaultního jazyka, jsou ve stejném pořadí načteny i balíčky z defaultního jazyka.

6.2.4 Konfigurace služby i18n

Konfigurace této služby se provádí v konfiguračním souboru *config.neon*, kde lze nastavovat tyto atributy:

- ***defaultLang*** - defaultní jazyk aplikace
- ***supportedLang*** - podporované jazyky
- ***directory*** - cesta k adresáři s jazykovými balíčky
- ***loadDefaultLang*** - zda hledat překlad i v defaultním jazyku, není-li k dispozici v aktuálně používaném jazyku
- ***extra*** - extra jazykové balíčky společné pro všechny jazyky (například formátovací řetězce pro výpis data, název aplikace aj.)

6.3 DataTables

Dále bylo implementováno rozhraní pro server-side podporu jQuery plug-inu DataTables, který je popisován v kapitole: 5.2.2.3 DataTables.

Toto rozhraní prozatím implementuje jedinou funkci *getTableData*.

Tato funkce umí zpracovat požadavek zasílaný z JavaScriptového plug-inu DataTables, na jehož základě obstará a vrátí data v požadovaném formátu z MySQL databáze.

Vstupními parametry této funkce jsou:

1. Název MySQL tabulky, ze které budou data čerpány.
2. Výčet sloupců tabulky, jejichž obsah bude zaslán plug-inu DataTables.
3. Požadavek zaslaný plug-inem DataTables.

7. Testování aplikace

Pro testování aplikace byl zvolen performance test filtru kontaktů. Důvodem byly počáteční velké problémy s časovou složitostí tohoto procesu. Původně se počítalo s filtrováním kontaktů v jazyce PHP, které mělo být provedené ihned po stažení kontaktů.

Výsledné filtrování i při vysoké optimalizaci na výkon trvalo řádově hodiny. Zajímavé přitom je, že vytížení serveru se při filtrování kontaktů pohybovalo kolem 10%.

Tento problém pravděpodobně pramenil z vlastností, nebo konfigurace použitého Apache, který hlídá a přiděluje prostředky jednotlivým PHP skriptům běžícím na serveru a má proto nastavené určité limity zamezující, aby jedno vlákno PHP skriptu vyčerpalo většinu použitelných zdrojů.

Z tohoto důvodu bylo filtrování kontaktů přepsáno do jazyka Java a provedeno testování výkonu filtru kontaktů. Výsledky tohoto testování lze shlédnout v na grafech v kapitole Příloha 2.

7.1 Návrh testu

Aplikace pro filtrování kontaktů bude upravena tak, aby před spuštěním filtrování dat bylo možno nastavit počet vláken pro paralelizaci procesu.

Dále bude při filtrování kontaktů počítána průměrná doba, zpracování jednoho kontaktu a u každého desátého kontaktu dojde k vypsání této hodnoty.

Jelikož jednotlivé sady kontaktů obsahují různý počet kontaktů, bude filtrování ukončeno vždy po zpracování 5 000 kontaktu.

7.2 Výsledky testu

Jednotlivé grafy (Figure 1, Figure 2, Figure 3 a Figure 4) reprezentují závislost objemu vstupních dat, na době zpracování jednoho kontaktu. Ze všech uvedených grafů vyplívá, že velikost testovací sady nehraje skoro žádnou roli. Důležitým aspektem je pouze to, jak moc chybných kontaktů se v daném setu nachází.

Co se týká paralelizace procesu, tak z Figure 5 (který reprezentuje průměrnou dobu zpracování kontaktu pro daný počet vláken) lze vidět, že pomocí paralelizace procesu lze dosáhnout značně lepších výsledků.

U jedno-vláknové aplikace se průměrná doba zpracování jednoho kontaktu pohybovala kolem 0,3s. Dle nefunkčních požadavků lze počítat s tím, že maximálním počet zaslaných

kontaktů nebude větší, jak 11 000. Jednoduchým výpočtem tak zjistíme, že doba zpracování takového množství by pak neměla trvat déle, jak 1 hodinu.

Při použití čtyř vláken se průměrné zpracování jednoho kontaktu pohybuje kolem 0,14s, z čehož dostáváme, že maximální doba zpracování 11 000 kontaktů by neměla trvat déle, než 26 minut.

Při použití šesti vláken se filtrování ještě zlepší, nicméně toto zlepšení již není natolik veliké, aby dobu zpracování nějak závratně ovlivnilo.

7.3 Závěr

Pro účely CSS bude pro filtrování kontaktů využita možnost paralelního zpracování dat a bude použito celkem čtyř vláken.

7.4 Další provedené testy

Další testy systému CSS rovněž prováděl ve své bakalářské práci Filip Šamánek. Zaměřil se přitom na testování funkcionality presenterů za pomocí nástroje Nette Tester a dále prováděl testování aplikace s uživateli.

8. Závěr

V této práci byly podrobně analyzovány všechny požadavky na výslednou funkcionalitu systému CSS obdržené od Firmy B a zadavatele práce Firmy A.

Na základě rozboru požadované funkcionality pak byla provedena analýza možných technologií a existujících řešení, které by se mohli uplatnit při vývoji této aplikace.

Z těchto technologií pak byly vybrány ty, jež plnily všechny funkční i nefunkční požadavky na systém CSS a zároveň se dali vhodně kombinovat, díky čemuž se pak usnadnil návrh konečného řešení a urychlily celkový vývoj této aplikace.

První verze aplikace byla spuštěna 1. 7. 2013 a od té doby do dnešního dne funguje bez závažnějších problémů.

Aplikace byla od prvního spuštění již několikrát vylepšena, přibyly například další generované statistiky a byla zcela přepsána aplikace pro generování SPSS reportů.

Navíc neustále vylepšujeme systém zabezpečení a zvyšujeme robustnost celé aplikace.

8.1 Návrhy na další vylepšení

Možným dalším vylepšením aplikace by mohlo být například implementování GUI pro snadnou tvorbu nových a editaci stávajících dotazníků, nebo vytvoření systému pro monitoring chodu call centra ve smyslu zobrazení informací o stavu klíčových funkcí CSS.

Dalším možným rozšířením, které chceme v brzké době implementovat je propojení CSS se systémem JIRA, který slouží ke komunikaci se zákazníkem a k hlášení chyb systému.

Zdroje

[1] Využití webových služeb a protokolu SOAP při komunikaci. *Jiří Kosek* [online]. [cit. 2014-05-19]. Dostupné z: http://www.kosek.cz/diplomka/html/websluzby.html

[2] Simple XLSX: Parse and retrieve data from Excel XLS files. *PHP classes* [online]. [cit. 2014-03-12]. Dostupné z: http://www.phpclasses.org/package/6279-PHP-Parse-and-retrieve-data-from-Excel-XLS-files.html

[3] *PHPExcel* [online]. [cit. 2014-03-12]. Dostupné z: http://phpexcel.codeplex.com

[4] *Doctrine Project* [online]. [cit. 2014-02-23]. Dostupné z: http://www.doctrine-project.org

[5] LightOrm - performance test. *SourceForge* [online]. [cit. 2014-02-23]. Dostupné z: http://sourceforge.net/apps/trac/phplightorm/wiki/LightOrm%20vs%20Propel%20vs%20Doctrine%20benchmark

[6] Databáze & ORM. *Nette Framework* [online]. [cit. 23-02-23]. Dostupné z: http://doc.nette.org/cs/2.0/database

[7] *Dibi* [online]. [cit. 2014-02-23]. Dostupné z: http://dibiphp.com/cs/

[8] Java SPSS library. *pmStation* [online]. [cit. 2014-03-25]. Dostupné z: http://spss.pmstation.com/spssw_index.jsp

[9] XML to SPSS Converter. *Source Forge* [online]. [cit. 2014-03-27]. Dostupné z: http://sourceforge.net/projects/xml2sav/

[10] IBM. In: *Input/Outpu Module*. Dostupné také z: http://www.docs.is.ed.ac.uk/skills/documents/3663/SPSSInput-OutputModule.pdf

[11] NYMAN, Jens. JForm - php form engine. *Source Forge* [online]. [cit. 2014-04-12]. Dostupné z: http://jformphp.sourceforge.net

[12] GRUDL, David. Formuláře. *Nette Framework* [online]. [cit. 2014-04-20]. Dostupné z: http://doc.nette.org/cs/2.0/forms

[13] FORMSTACK, LLC. *Formstack* [online]. [cit. 2014-04-20]. Dostupné z: https://www.formstack.com

[14] LLC, Interlogy. *JotForm* [online]. [cit. 2014-04-20]. Dostupné z: http://www.jotform.com

[15] SOFTWARE, Appnitro. *MachForm* [online]. [cit. 2014-04-20]. Dostupné z: http://www.appnitro.com

[16] *Nette Framework* [online]. [cit. 2014-02-21]. Dostupné z: http://nette.org/cs/

[17] Velký test PHP frameworků: Zend, Nette, PHP a RoR. *ROOT.CZ* [online]. [cit. 2014-02-27]. Dostupné z: http://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/

[18] Core Admin. *WrapBootstrap* [online]. [cit. 2014-02-17]. Dostupné z: https://wrapbootstrap.com/theme/core-admin-WB0135486

[19] *Bootstrap* [online]. [cit. 2014-05-12]. Dostupné z: http://getbootstrap.com

[20] *DataTables* [online]. [cit. 2014-05-15]. Dostupné z: https://datatables.net

[21] Přihlašování & oprávnění uživatelů. *Nette Framework* [online]. [cit. 2014-03-12]. Dostupné z: http://doc.nette.org/cs/2.1/access-control#toc-permission-acl

[22] ZANDSTRA, Matt. *PHP Objects, Patterns, and Practice*. Apress, 26112013. ISBN-13: 9781430260318.

[23] SCHWARTZ, Baron, ZAITSEV, Peter a TKACHENKO, Vadim. *High Performance MySQL: Optimization, Backups, and Replication*. O'Reilly Media, 30032012. ISBN-13: 9781449314286.

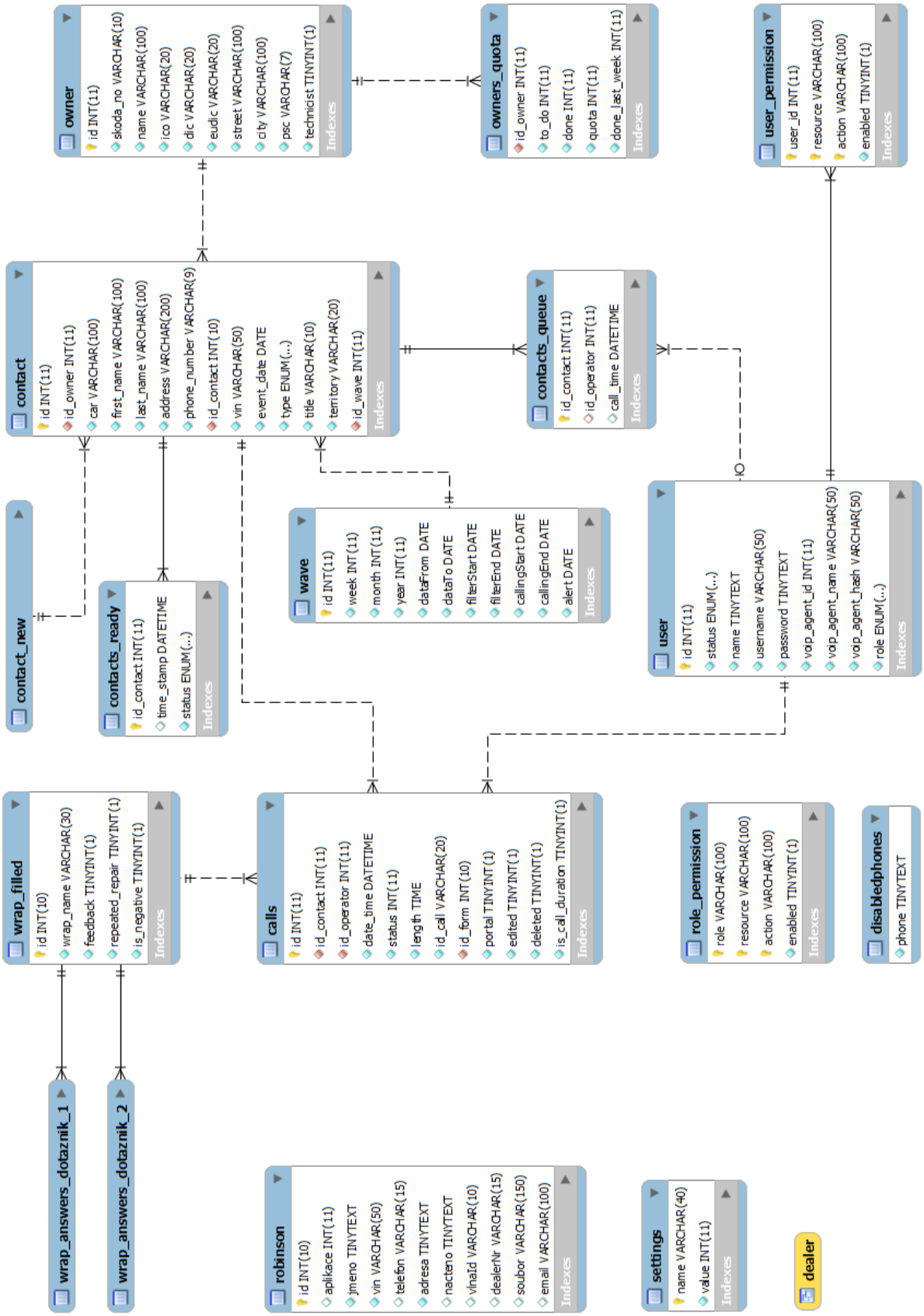
[24] GEORGE, Darren a MALLERY, Paul. *IBM Spss Statistics 21 Step by Step: A Simple Guide and Reference*. Prentice Hall College Div, 16072013. ISBN-13: 9780205985517.

Seznam obrázků

Seznam grafů

Seznam tabulek

Příloha 1



Obrázek 5 - Návrh databáze systému CSS

Příloha 2

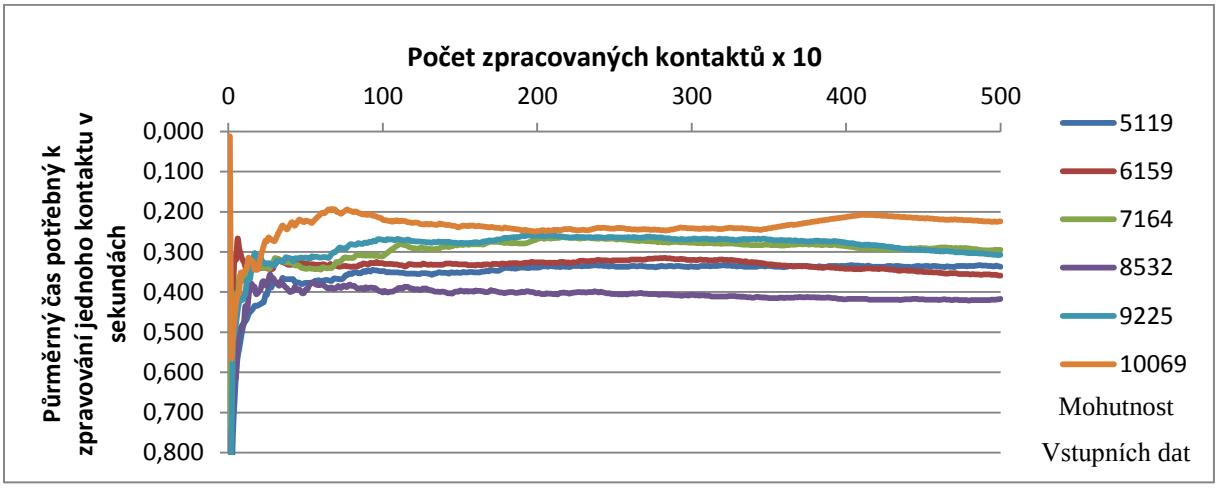


Figure 1 - Test provedený s 1 vláknem

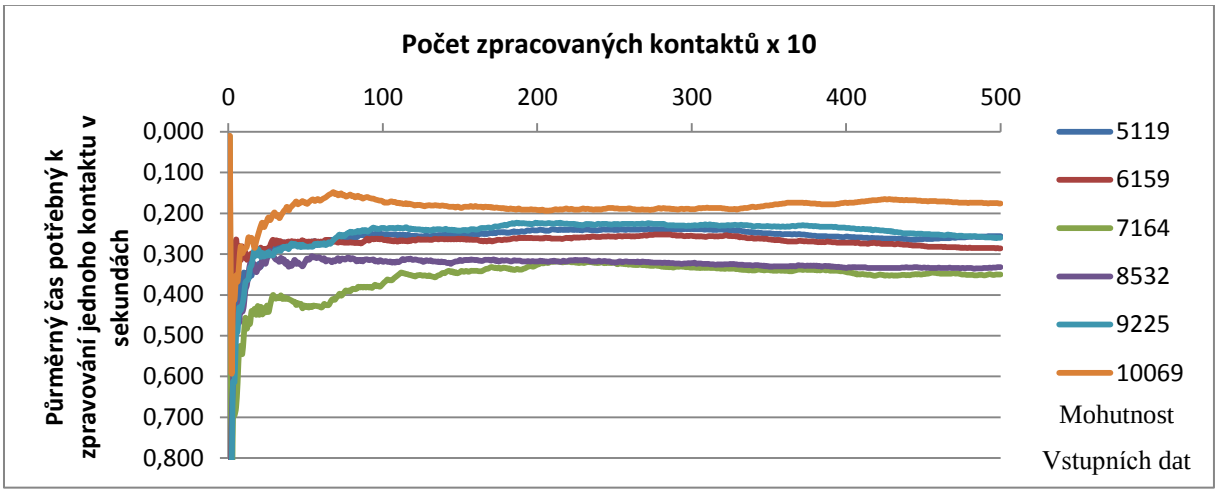


Figure 2 - Test provedený s 2 vlákny

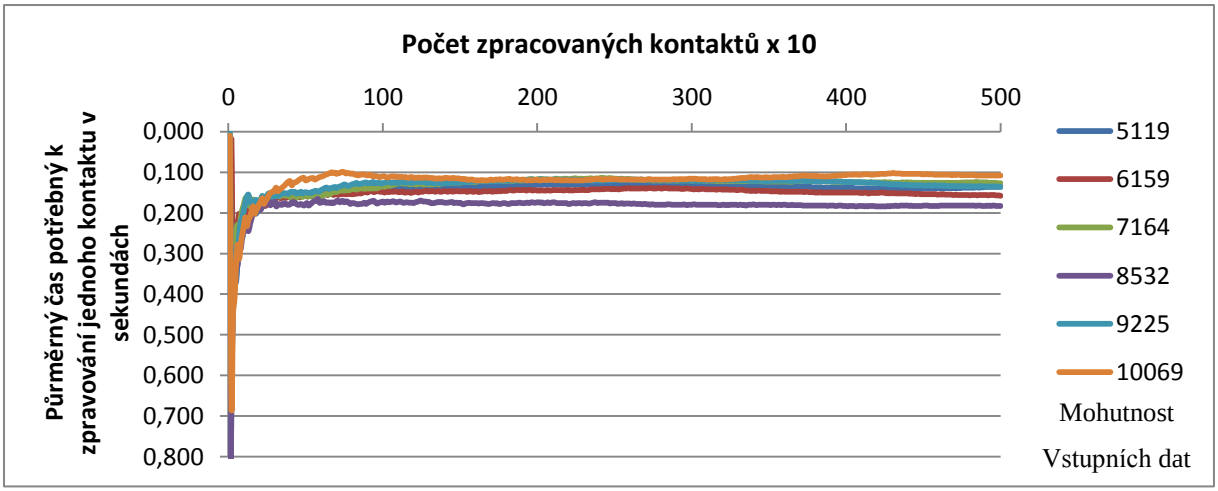


Figure 3 - Test provedený s 4 vlákny

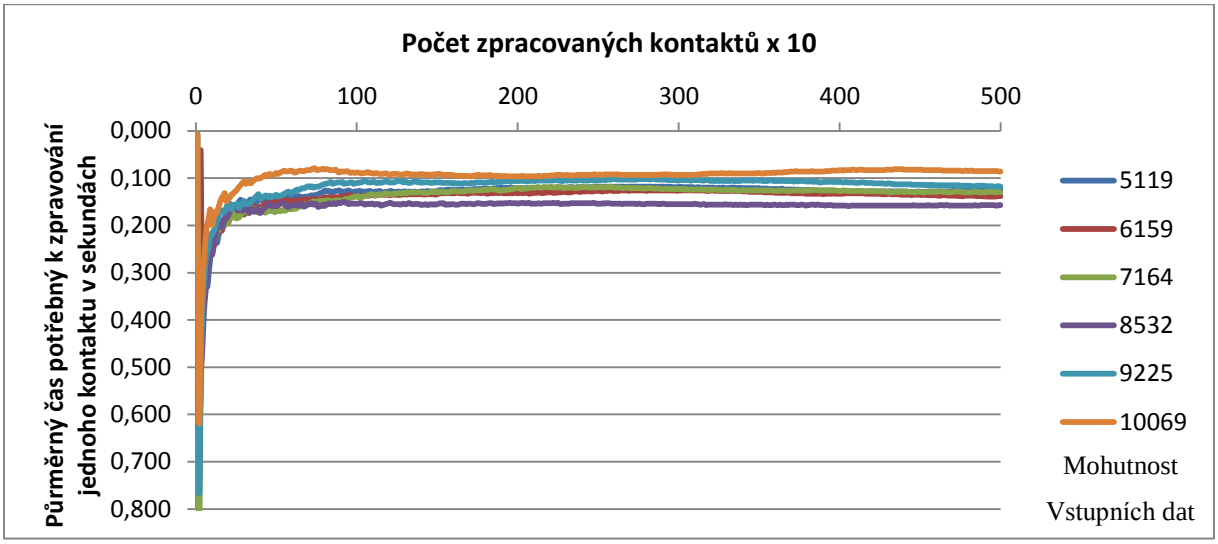


Figure 4 - Test provedený s 6 vlákny

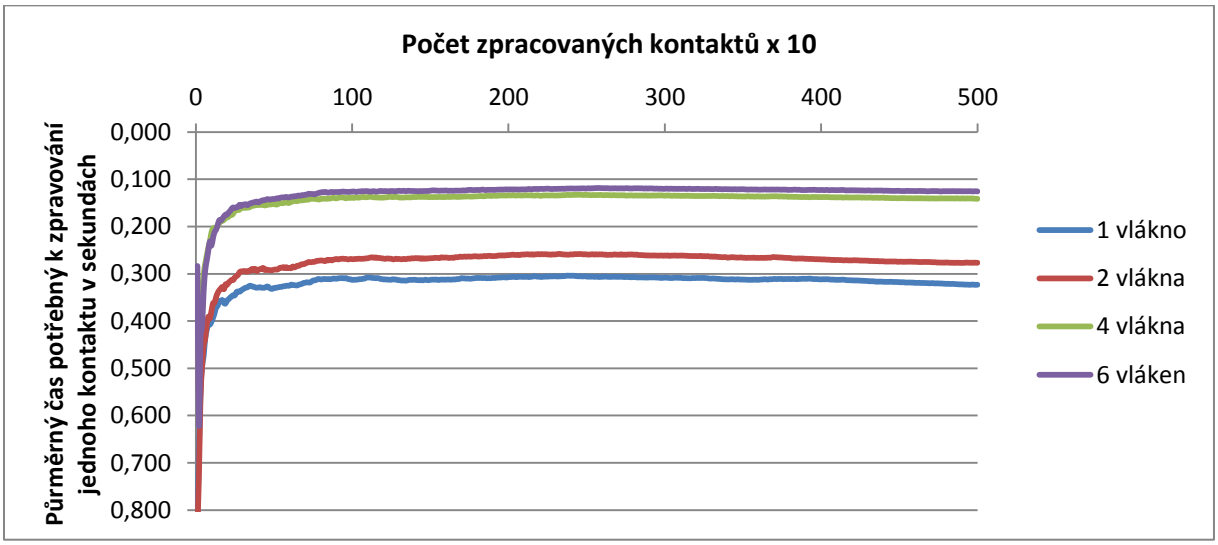


Figure 5 - Výsledné vyhodnocení testu