Mendelova univerzita v Brně
Provozně ekonomická fakulta

Webová aplikace pro tvorbu interaktivních formulářů

Diplomová práce

Vedoucí práce:
Ing. Jiří Lýsek, Ph.D.

Bc. Roman Koláček

Brno 2017

Na tomto místě bych chtěl poděkovat svému vedoucímu diplomové práce Ing. Jiřímu Lýskovi, Ph.D. za odborné vedení práce, podporu, připomínky a hlavně trpělivost při jejím vytváření.

Čestné prohlášení

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

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

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

V Brně dne 2. ledna 2017 _____

Abstract

Koláček, R. Web application for interactive form assembly. Diploma thesis. Brno, 2017.

Thesis is focused on analysis, design and implementation of web application facilitating the assembly of interactive forms. Feature of this application is also administration of filled-in forms and registered users. Thesis also encompases analysis of comparable simillar applications. Practical part of thesis is final implementation of web appllication using the Nette framework along with SQL database.

Keywords

Forms, Nette framework, web application, PHP, MySQL.

Abstrakt

Koláček, R. Webová aplikace pro tvorbu interaktivních formulářů. Diplomová práce. Brno, 2017.

Tato práce se zabývá analýzou, návrhem a implementací webové aplikace pro tvorbu interaktivních formulářů. Součástí webové aplikace je i administrace vytvořených formulářů a registrovaných uživatelů. V práci jsou taktéž analyzovány současné konkurenční webové aplikace. Součástí práce je i finální implementace webové aplikace pomocí Nette frameworku s využitím databáze MySQL.

Klíčová slova

Formuláře, Nette framework, webová aplikace, PHP, MySQL.

Obsah

1 Úvod a cíl práce 17
1.1 Úvod ... 17
1.2 Cíl práce ... 17

2 PHP frameworky 18
2.1 Laravel ... 18
2.2 Symfony ... 19
2.3 Nette ... 19
2.4 CodeIgniter ... 20
2.5 Yii ... 21
2.6 Výsledné srovnání ... 21
2.7 Nette a jeho principy ... 22
2.7.1 MVC architektura ... 22
2.7.2 Latte šablony ... 23
2.7.3 Zabezpečení ... 24
2.7.4 Dependency Injection ... 24
2.7.5 Composer ... 25

3 Webdesign 26
3.1 Grafický design ... 26
3.2 Bootstrap ... 28
3.3 JavaScript ... 28
3.4 jQuery ... 29
3.5 FormBuilder ... 29

4 Metodika 31

5 Analýza konkurenčních řešení 32
5.1 Google formuláře ... 32
5.2 WuFoo ... 33
5.3 Typeform ... 34

5.4 Formstack 35
5.5 JotForm 37
5.6 Vyhodnocení analýzy konkurenčních formulářů 38
5.7 Specifikace na funkcionalitu formulářů 39

6 Návrh aplikace 40
6.1 Uživatelské role 40
6.2 Správa veřejného modulu 41
6.3 Správa formulářů 41
6.4 Správa odpovědí 42
6.5 Správa příspěvků 43
6.6 Správa administrace 44
6.7 Grafický návrh 45
6.8 TwiGrid tabulky 47
6.9 Návrh databáze 47

7 Implementace s využitím Nette frameworku 53
7.1 Adresářová struktura projektu 53
7.2 Uživatelské role 54
7.3 Registrace uživatele 55
7.4 FormBuilder 57
7.5 Uložení vytvořeného formuláře 60
7.6 Vykreslení vytvořeného formuláře 60
7.7 Další nastavení vytvořeného formuláře 62
7.8 Nová jazyková verze formuláře 62
7.9 Sdílení formuláře 63
7.10 Odpovědi na formulář 64
7.11 Zobrazení vytvořených formulářů 65
7.12 Zobrazení odpovědí na formulář 66
7.13 Grafické vyhodnocení 67
7.14 Statistiky 69
7.15 Novinky 70
7.16 Komentáře 71

7.17 Prémiový přístup 72
7.18 Administrace 73
7.19 Správa uživatelů 73
7.20 Přidání recenze 73
7.21 Přidání novinek 74
7.22 Přehled plateb 74
7.23 Vícejazyčnost webové aplikace 75
7.24 Přidání domén 75

8 Technické a ekonomické vyhodnocení 77

9 Závěr a diskuze 79

10 Literatura 81

A ERD návrh databáze 84

B Ekonomické vyhodnocení bez reklamy 85

C Ekonomické vyhodnocení s reklamou 86

Seznam obrázků

Obr. 1 Ukázka responzivního designu 27

Obr. 2 Ukázka tvorby formuláře v základní verzi FormBuilderu 30

Obr. 3 Ukázka tvorby formuláře pomocí webové aplikace Google Forms 33

Obr. 4 Ukázka tvorby formuláře pomocí webové aplikace WuFoo 34

Obr. 5 Ukázka tvorby formuláře pomocí webové aplikace TypeForm 35

Obr. 6 Ukázka tvorby formuláře pomocí webové aplikace Formstack 36

Obr. 7 Ukázka tvorby formuláře pomocí webové aplikace JotForm 38

Obr. 8 Případy užití základní rozvržení 40

Obr. 9 Případy užití nepřihlášeného uživatele 41

Obr. 10 Případy užití správy formulářů 42

Obr. 11 Případy užití správy odpovědí 43

Obr. 12 Případy užití správy příspěvků 44

Obr. 13 Případy užití správy administrace 45

Obr. 14 Ukázka informační části webu pro nepřihlášeného uživatele 46

Obr. 15 ERD s tabulkami týkajícími se uživatele 48

Obr. 16 ERD s tabulkami týkajícími se formulářů 49

Obr. 17 ERD s tabulkami týkajícími se odpovědí na formulář 51

Obr. 18 ERD s tabulkami týkajícími se novinek, domén a komentářů 52

Obr. 19 Adresářová struktura projektu 54

Obr. 20 Ukázka tvorby formuláře v upravené verzi FormBuilderu 60

Obr. 21 Ukázka překladu formuláře do češtiny 63

Obr. 22 Ukázka stránky, která se zobrazí po úspěšném odeslání formuláře 65

Obr. 23 Ukázka seznamu vytvořených formulářů 66

Obr. 24 Ukázka grafického vyhodnocení odpovědí 68

Obr. 25 Ukázka editace grafu 68

Obr. 26 Ukázka statistického vyhodnocení odpovědí 70

1 Úvod a cíl práce

1.1 Úvod

Dotazníková forma výzkumu je jednou z nejrozšířenějších a poměrně precizních metod k získávání informací od cílových subjektů. Principy sestavování dotazníků musí plně reflektovat požadovanou přesnost údajů v návaznosti na pozdější kvantifikovatelné vyjádření získaných dat. Tudíž při vývoji aplikace pro sestavování dotazníkových formulářů je nutno tyto principy respektovat a zároveň eliminovat zbytečné komplikace v interpretaci výsledků a datový šum způsobený nejednoznačně interpretovanými odpověďmi v rámci pokládaných otázek. Proto samotná aplikace sloužící k vytváření dotazníkových formulářů musí uživateli poskytovat intuitivní ovládání i určitou volnost modifikace jednotlivých šablon a uživatelský komfort při sestavování konečných formulářů, ale zároveň taktéž reflektovat zavedené principy sestavování dotazníků. Navrhovaná aplikace by měla být navrhnuta tak, aby výsledný proces sestavování dotazníků byl pro uživatele co nejméně časově náročný a složitý. Snaha o zavedení šablonovitého modelu přispívá ke zlepšení efektivity sestavování dotazníku a taktéž jeho parametrickému a statistickému vyhodnocování. Velký důraz je kladen na zjednodušení a urychlení procesů jednoduchou mechanickou manipulací drag&drop, kde si uživatel sám z předdefinovaných prvků vytvoří dotazník podle svých představ.

1.2 Cíl práce

Cílem práce je vytvoření návrhu webové aplikace na tvorbu formulářů a následně ji naimplementovat na základě stanovených požadavků.

Webová aplikace umožní uživatelsky jednoduchou a intuitivní tvorbu formulářů, zaznamenávání odpovědí na dané formuláře a přehledné zobrazování odpovědí, a to nejen v textové, ale i grafické podobě.

2 PHP frameworky

Pro tvorbu webových aplikací, zaměříme-li se primárně na jazyk PHP, ve kterém bude výsledná aplikace naprogramována, je v současné době možné využít buď čistě jazyk PHP, nebo využít mnoha PHP frameworků, které programátorovi v mnoha ohledech velmi usnadní a urychlí práci.

Framework je definován jako nástroj, který slouží pro vývoj softwarových aplikací pro danou platformu. S jeho využitím může vývojář efektivněji, přehledněji a bezpečněji vyvíjet aplikace, neboť framework již obsahuje předdefinované třídy a funkce, díky čemuž vývojář nemusí znovu vymýšlet funkcionalitu, ale místo toho již tyto vytvořené třídy a funkce použít ve svých projektech. Framework je svým způsobem velmi podobný aplikačnímu programovacímu rozhraní (API), avšak technicky vzato sám framework obsahuje vlastní API, které poskytuje přístup k vlastním třídám a funkcím. (The Tech Terms Computer Dictionary, 2013, [online])

V této analýze je vycházeno z průzkumů, které byly uskutečněny v roce 2015 serverem SitePoint (SitePoint, 2015, [online]) a Noeticforce (Noeticeforce, 2016, [online]), a zároveň toto hodnocení je z důvodu objektivnosti porovnáno s hodnocením na GitHubu, které vyjadřuje míru zájmu uživatelů. Analyzováno je pět frameworků, které v těchto průzkumech dopadly nejlépe.

2.1 Laravel

Laravel je open source PHP framework, který je vydávaný pod licencí MIT. První verze byla vydána teprve v roce 2012, čímž se stává jedním z nejmladších frameworků, ale to mu nezabránilo v tom, aby se stal jedním z nejpoužívanějších frameworků na trhu vůbec. Přestože se Laravel objevil mezi frameworky velmi pozdě, využil tuto nevýhodu ve svou výhodu – vyhnul se problémům, chybám a nedostatkům již existujících frameworků. Již od začátku začal stavět na základech Symfony, díky čemuž se stal robustním a komponentně založeným frameworkem. Laravel je v současné době ve verzi 5.3 a i přes svoji krátkou existenci získal ve výše prováděném průzkumu serveru SitePoint první místo a na GitHubu získal 26 486 hvězd, což je nejvíce ze všech čtyř porovnávaných frameworků.

Tento framework využívá MVC (Model-View-Controller) architektury. Jeho výhodou je, že je zvláště optimalizovaný pro použití v reálných situacích, neboť disponuje často používanými procedurami, nutnými při vývoji webových aplikací. Taktéž využívá objektově orientované programování. (Bean, 2015)

Mezi základní funkcionality tohoto frameworku patří modularita, kdy Laravel využívá více než 20 odlišných komponent a spojuje je do individuálních modulů. Tyto moduly jsou propojeny s Composerem, který zajišťuje jejich závislosti a aktualizace. Dále umožňuje přehledné a flexibilní routování, které má na starost správu směrování a zpracování dotazů na jednom místě. Obsahuje taktéž veškeré nástroje pro komunikaci a práci s databází, kdy místo ručního psaní SQL příkazů lze využít metody, které se postarají o sestavení samotného SQL

příkazu. Laravel využívá šablonovací systému Blade, který umožňuje vytvářet přehledné hierarchické layouty s předdefinovanými bloky. Tento šablonovací systém dynamicky vkládá obsah za použití vlastního způsobu zápisu, který je následně překládán do PHP zdrojového kódu. Taktéž díky knihovně SwiftMailer od Symfony zvládá jednoduché posílání emailů. Nechybí zde ani správa sessions nebo autentizace, kontrolující přístup uživatelů, dokonce i zasílání zapomenutého hesla uživatelům. Za zmínku taktéž stojí příkazová řádka Artisan, ze které lze Laravel snadno ovládat a lze pomocí něj pracovat se systémem migrace databáze – tedy umožňuje upravovat databázovou strukturu, aniž by to ohrozilo chod aplikace i samotné databáze. Pro práci s databází je využit ORM Eloquent, který poskytuje jednoduchou implementaci návrhového vzoru Active Record.

Pro testování aplikace využívá integrované knihovny PHPUnit. (Laravel, [online])

2.2 Symfony

Symfony je stejně jako Laravel open source framework vydávaný pod licencí MIT. Podle hodnocení na GitHubu dostal celkem 13 096 hvězd, tedy druhý nejvyšší počet z analyzovaných frameworků. Počátky Symfony sahají až do roku 2005. Poslední aktuální verze tohoto frameworku je 3.1 a v současné době se skládá z celkem 34 oddělených a znovupoužitelných komponent, které tvoří samotný základ celé řady známých aplikací, jako například Drupal, eZ Publish, Joomla (systémy pro správu obsahu), phpBB (systém pro správu diskusních fór) a mnoha dalších. Tyto komponenty lze nainstalovat pomocí Composeru, čímž je opět zajištěna jejich závislost a aktuálnost.

Architektura Symfony je založena na MVC architektuře a objektově orientovaném modelu. Symfony využívá pouze jeden typ routování, kdy URL adresa není ve tvaru např. *index.php?article_id=1*, ale zápis URL adresy je namísto toho ve tvaru */read/some-article-name*. Pro vykreslování stránek se v Symfony používá šablonovací systém Twig, který pro úpravu výstupů využívá vlastní způsoby zápisu, které se následně překládají do PHP jazyka. Stejně jako Laravel využívá pro testování integrované knihovny PHPUnit. Pomocí konfiguračního souboru zvládá i autorizaci a autentizaci uživatelů, stejně tak je řešena i lokalizace a nastavení poštovního serveru SwiftMailer. (Symfony, [online])

Pro práci s databází je využito ORM postavené na Doctrine za použití vlastního dotazovacího jazyka DQL (Doctrine Query Language), který je inspirovaný podobným dotazovacím jazykem HQL (Hibernate Query Language) z objektově relačního frameworku Hibernate pro programovací jazyk Java. (Doctrine Project, [online])

2.3 Nette

Nette je český open source framework, jenž cílí především na rychlost a bezpečnost. Tento framework, licencovaný pod dvěma licencemi, a to buď

GNU GPL, nebo New BSD. V současné době je poslední stabilní verze 2.4, kterou vyvíjí česká komunita, tudíž má oproti ostatním frameworkům dokumentaci kromě angličtiny i v českém jazyce. Taktéž tento framework disponuje velkou českou komunitou na fóru, kde lze mnoho informací, které v dokumentaci nejsou dostatečně vysvětleny, vyhledat. Nevýhodou oproti ostatním frameworkům je menší komunita, která tento framework aktivně vyvíjí, ale i přesto je podle SitePoint třetím nejrozšířenějším frameworkem vůbec (především v České republice), avšak podle hodnocení na GitHubu získal pouze 1 158 hvězd, což je nejméně ze všech analyzovaných frameworků.

Opět je zde využívána MVC architektura (v tomto případě spíše MVP, neboť pro Controller se v Nette používá název Presenter) a objektově orientovaný návrh. Zároveň má snahu o čistý a jednoduchý kód, který vede k dobrým programovacím návykům a má strmou křivku učení. Například tvorba formulářů je v Nette mnohem přehlednější a jednodušší než u ostatních frameworků.

Kromě vysoké úrovně zabezpečení, které eliminuje výskyt bezpečnostních děr a jejich zneužití, obsahuje Nette velmi dobře navržený ladicí nástroj Tracy (v české komunitě se spíše vžil název Laděnka), který programátorovi umožňuje zavčas během vývoje aplikace odhalit a opravit chyby, tyto chyby logovat a dále poradí jak vypisovat proměnné. Taktéž v Debugger Baru zobrazuje, kolik času bylo potřeba na vygenerování samotné stránky, kolik dotazů proběhlo na databázi (opět spolu s časem, který tyto dotazy potřebovaly pro získání dat z databáze) a jednoduchou routovací tabulku. Nette dále zvládá efektivně komunikovat s databází pomocí metod, tudíž programátor nemusí vypisovat celý SQL příkaz, ale Nette si jej stejně jako Laravel sám sestaví, avšak nejedná se o plnohodnotné ORM. Nette využívá pro komunikaci s databází komponentu Database, případně lze využít volně dostupnou databázovou knihovnu Dibi.

Pro vykreslování šablon využívá šablonovací systém Latte, který taktéž vývojáři ulehčí práci a zabezpečí výstup před zranitelnostmi jako je XSS. Využívá vlastní makra a bloky, které jsou následně, stejně jakou u výše uvedených frameworků, překládány do PHP zdrojového kódu.

Mezi společnosti, které na vývoj webových aplikací využívají Nette framework, patří například GE Money, Mladá fronta, Vltava-Labe-Press, DHL, ESET, Slevomat a další. (Nette framework, [online])

2.4 CodeIgniter

CodeIgniter je i přes svoji velikost (2 MB včetně dokumentace) velmi výkonný PHP framework. Vznikl v roce 2006 a je vyvíjen především pro vývojáře, kteří potřebují jednoduchý a elegantní nástroj pro tvorbu plnohodnotných webových aplikací. Jedná se opět o open source projekt s Apache/BSD licencí, jehož poslední stabilní verze je 3.1.0. V současné době má na GitHubu 13 152 hvězd, což je téměř stejně jako u Symfony.

Přestože využívá MVC architekturu, nenutí vývojáře ji používat. Stejně tak nevyžaduje žádné speciální pravidla kódování a využívání konvencí, vše nechává na vývojáři. Na rozdíl od výše uvedených frameworků, CodeIgniter nepoužívá

žádný vlastní jazyk v šablonovacím systému. Velkou výhodou tohoto frameworku je to, že jej není potřeba téměř vůbec konfigurovat, neboť všechna důležitá nastavení jsou již obsažena v instalačním balíčku. Stejně tak využívá knihovny s jednoduchým rozhraním, které řeší časté a opakující se úlohy, čímž opět šetří vývojářům čas. (CodeIgniter Web Framework, [online])

2.5 Yii

Poslední analyzovaný framework je Yii. Začal být vyvíjen v lednu 2008 jako open source pod licencí BSD. Vychází ze zkušeností z vývoje Prado frameworku, takže se podobně jako Laravel mohl inspirovat v jeho silných stránkách, a zároveň se mohl vyhnout chybám a nedostatkům, které vznikaly při jeho vývoji. Díky těmto zkušenostem trvalo vydání první stabilní verze velmi krátce (první verze byla vydána již v prosinci téhož roku). Protože byl Yii framework již od začátků dostatečně rychlý, bezpečný a profesionální framework, získal si záhy oblibu u vývojářů. V říjnu 2014 vyšla verze Yii 2, která byla kompletně přepsána z původní verze Yii. V této verzi byly využity nejnovější technologie a vylepšení, díky čemuž se stal framework ještě lepším.

Stejně jako všechny výše uvedené frameworky, i Yii využívá MVC architekturu, čímž zajišťuje oddělení datové, logické a prezentační vrstvy. Aby byl tento framework dostatečně rychlý, načítá pouze to, co je pro vývojáře potřeba, má výkonnou podporu cachování a je taktéž přímo navržen tak, aby pracoval efektivně s AJAXem. Kromě rychlosti má snahu o co největší bezpečnost. Proto již v sobě zahrnuje vstupní validaci, filtruje odchozí data, chrání před SQL injection a taky proti Cross-site scripting.

Yii dále podporuje přístup k různým databázím pomocí DAO (Database Access Objects) a Active Record.

2.6 Výsledné srovnání

Vyhodnocení výše analyzovaných frameworků je poměrně relativní, protože Symfony má sice nižší hodnocení od uživatelů než Laravel, ale přitom z něj Laravel vychází. Zároveň zde dochází k určitému zkreslení, především u Nette frameworku. Při srovnání počtu hlasů podle zemí, byla překvapivě Česká republika se 770 hlasy na druhém místě hned za USA (celkem 819 hlasů) v celkovém počtu hlasování. To výrazně posunulo tento českou komunitou vyvíjený framework do popředí. Z České republiky hlasovalo celkem 83 % Čechů právě pro Nette framework. Proto je lepší vycházet z hodnocení na GitHubu, který je přesnější, neboť počet hodnotících je mnohonásobně vyšší a výsledky hodnocení jsou tedy i méně zkreslené. I přesto, až tedy na výše zmíněný Nette framework, jsou jak hodnocení na stránkách SitePoint, tak na GitHubu velmi podobné.

Co se týče frameworku, který bude použit pro vývoj aplikace, nejvíce mě zaujal především Nette framework, přestože není oproti ostatním frameworkům příliš rozšířen ve světě. Hlavní důvod je ten, že se velká část aplikace bude zabý-

vat právě tvorbou formulářů. V některých případech může být kód, který bude řešit vykreslování formulářů, velmi obsáhlý a bude mít spoustu podmínek pro jejich správnou tvorbu. Nette framework si zakládá na jednoduché tvorbě formulářů a jejich znovupoužitelnosti, navíc i samotný zdrojový kód při tvorbě formulářů je mnohem jednodušší, vyžaduje kratší zápis a je tudíž přehlednější. Přitom zde není potřeba řešit celou řadu rutinních úkolů, jako je psaní dvojí validace (na straně serveru a klienta), neboť to všechno řeší tento framework. Navíc má tento framework velmi dobré zabezpečení formulářů a nevyžaduje manuálně nic nastavovat. Chrání aplikace před útokem Cross Site Scripting (XSS) i Cross-Site Request Forgery (CSRF), odfiltruje ze vstupů kontrolní znaky, ověří validitu UTF-8 kódování nebo jestli nejsou položky vybrané v select boxech podvržené atd.

Kromě toho jako jeho další výhodu oproti konkurenčním frameworkům shledávám výborný ladicí nástroj Tracy, který je velkým pomocníkem při vývoji. Další důvod je i ten, že Nette framework disponuje obsáhlou dokumentací nejen v angličtině, ale i v češtině. Navíc má podporu velké české komunity na fóru, kde jsou rozebírány a vysvětlovány i méně časté problémy týkající se práce s Nette frameworkem, a to mnohdy ještě podrobněji než v samotné dokumentaci.

2.7 Nette a jeho principy

Jelikož byl na základě analýzy vybrán Nette framework, budou ještě detailněji rozepsány jeho principy a vlastnosti.

2.7.1 MVC architektura

Je to softwarový designový vzor, postavený za účelem propojení tří základních typů komponent, používaný především v programovacích jazycích se silným zaměřením na objektově orientované programovací softwarová paradigmata. Tyto tři typy komponent jsou nazývány jako *model, view a controller*.

Model zajišťuje práci s databází. Jakákoliv akce uživatele, jako je například přihlášení, změna hodnoty v databázi atd, představuje akci modelu. Model neví o existenci view ani kontroleru. Všechen kód, který má za úkol komunikovat s databází, by měl být právě v modelu.

View, neboli pohled, zajišťuje správné zobrazení dat uživateli. To může zahrnovat HTML značkovací jazyk, CSS styly, JavaScriptové soubory atd. Celkově cokoliv, co uživatel vidí, nebo s čím může interagovat, je uloženo v pohledu. Někdy je to, co uživatel vidí, kombinací i několika různých pohledů v téže žádosti.

Controller, neboli řadič, zajišťuje propojení modelu a pohledu mezi sebou. Odděluje logiku modelu od pohledu za účelem zpracování dat z modelu a jejich poskytnutí uživateli prostřednictvím pohledu. Zpracovává požadavky uživatele, na jejichž základě volá model a poté žádá view o vykreslení dat. V Nette frameworku jsou presentery, které jsou obdobou controlleru. (Pitt, 2012)

2.7.2 Latte šablony

Latte je šablonovací systém, který ušetří a zjednoduší práci, zároveň zabezpečuje výstup před zranitelnostmi, jakými je například XSS.

Oproti PHP má mnohem jednodušší zápis. Používá dva druhy speciálních značek. Makra, která se zapisují pomocí složených závorek – např. `{foreach $items as $item} … {/foreach}`

Navíc každé párové makro se dá přepsat do tzv. n:makra, která se používají přímo v HTML značkách jako jejich speciální atributy. Příkladem může být výše uvedené makro v následující podobě jako n:makro `<p n:foreach="$items as $item"> … </p>`, které má za úkol vypsat každou hodnotu na nový řádek.

V Latte není potřeba, jako například u PHP, volat funkci *htmlSpecialChars*, která by automaticky escapovala proměnné, neboť Latte se o escapování stará sám. Díky využívání technologie Context-Aware Escaping rozeznává, ve které části dokumentu se makro nachází, díky čemuž zvolí správné escapování.

Latte taktéž umožňuje na proměnné využít filtry (případně se ještě používá název modifikátory nebo podle starších verzí helpery). Stačí pouze za proměnnou napsat svislou čáru a za ní název filtru, který modifikuje proměnnou. Filtrů lze použít i více, v tomto případě by byly odděleny opět svislou čárou a jejich aplikace by probíhala postupně podle pořadí, v jakém jsou za sebou. V případě, že filtr potřebuje parametry, lze je zadat za jménem daného filtru oddělené dvojtečkami nebo čárkami. Například `<h1>{$heading|truncate:20,''}</h1>`

Přestože Latte používá svůj způsob zápisu kódu, je velice rychlé, neboť tento kód překládá do nativního PHP kódu, který ukládá do cache na disk. Proto má stejný výkon, jako kdyby byly šablony psány v čistém PHP.

V případě, že se změní zdrojový soubor, je šablona pokaždé znovu automaticky přeložena a uložena do cache na disku, takže pokud se během vývoje šablona edituje, jsou všechny změny okamžitě viditelné v prohlížeči.

Pokud vývojář udělá v šabloně jakoukoliv chybu, ať už překlep nebo volání proměnné, která nebyla šabloně poslána z presenteru, okamžitě o ní informuje Laděnka, která zobrazí zdrojový kód šablony a zvýrazní řádek, na kterém se nachází chyba. Tuto chybu většinou dostatečně srozumitelně vysvětluje chybová zpráva. Navíc Latte generuje HTML výstup se správným hierarchickým odsazováním značek. Výsledný zdrojový kód webu je poté mnohem přehlednější. (Nette framework, [online])

V Latte šablonách lze taktéž vkládat makra, která zajišťují vícejazyčnost webu. Nutné je však vytvořit obsah slovníku, který používá formátování *neon*, jehož zápis do češtiny a angličtiny by vypadal následovně pro slovník ui.cs_CZ.neon:

```
menu1: Vytvořit formulář
menu2: Moje formuláře
```

a pro a ui.en_EN.neon slovník:

```
menu1: Create form
menu2: My forms
```

V šabloně se poté již jenom odkazuje pomocí makra na konkrétní překlad. Nutné je dbát na přesný formát makra, tedy za levou závorkou vložit podtržítko s mezerou a napsat název slovníku spolu s názvem překládaného prvku oddělené tečkou.

```
<div>
    <p>{_ ui.menu1}</p>
    <p>{_ ui.menu2}</p>
</div>
```

V případě, že daný překlad v jazyce není definován, zobrazí se v šabloně namísto překladu název daného makra, např. při vložení nedefinovaného makra `{_ ui.menu3}` je zobrazeno ui.menu3, díky čemuž lze v případě testování překladů zjistit překlep v názvu makra, případně že nejsou všechny překlady definované ve slovnících.

2.7.3 Zabezpečení

Nette framework zabezpečuje výstup před zranitelnostmi, jako je XSS. Ač se jedná o jeden z nejtriviálnějších způsobů, jak zaútočit na webové stránky, jedná se o nejčastěji využívanou a závažnou zranitelnost. Aby se jí předešlo, je nutné důsledné dodržování escapování vypisovaných dat, tedy převod znaků, které mají v daném kontextu určitý specifický význam, na znaky jiné odpovídající sekvence. Aby se těmto bezpečnostním dírám předešlo, používá šablonovací systém Latte automatické escapování, které navíc rozeznává, ve které části dokumentu se makro nachází a podle toho používá takovou escapovací funkci, aby byly různé proměnné správně escapovány. Takže v případě, že se proměnná nachází uvnitř HTML, je použito jiné escapování této proměnné, než když se tato proměnná nachází uvnitř JavaScriptu a podobně. (Nette framework, [online])

2.7.4 Dependency Injection

Dependency Injection odebírá třídám zodpovědnost za získávání objektů, které potřebují ke své činnosti (tzv. služeb), a místo toho jim služby předává při vytváření. Dependency Injection se přímo stará o to, aby se třídám předalo právě to, co potřebují. Výsledkem tedy je, že kód neobsahuje žádné skryté závislosti a vše, co třída potřebuje ke své funkčnosti, jí je předáno z vnějšku pomocí konstruktoru, metody nebo vlastností. (Nette framework, [online])

2.7.5 Composer

Pro správu závislostí v PHP je využíván nástroj Composer, který dovoluje deklarovat libovolně složité závislosti jednotlivých knihoven, které poté nainstaluje do projektu. Composer lze stáhnout jako *.phar* soubor, případně použít instalátor, který má Composer na svých stránkách.

Composer je navíc propojený s verzovacím nástrojem Git, tudíž lze projekt vytvořit lehce pomocí jednoho příkazu, který naklonuje repozitář *nette/sandbox* přímo z GitHubu a následně okamžitě stáhne Nette framework. (Nette framework, [online])

3 Webdesign

3.1 Grafický design

V roce 2007 s příchodem zařízení iPhone na trh nastal zlom ve web designu, protože do té doby byl design webových stránek přizpůsoben pouze na velikost obrazovek monitorů. Dříve byly weby zobrazovány pouze na monitorech, jejichž úhlopříčka byla sice různá, ale design se dal ještě lehce přizpůsobit tak, aby web vypadal dobře jak na menších monitorech, tak i na větších – stačila fixní šířka odpovídající nejčastějšímu rozlišení monitorů té doby. Ovšem s příchodem prvních chytrých telefonů nastala otázka, jak navrhnout stránky tak, aby byly správně a vhodně zobrazeny i na malých obrazovkách, které tyto telefony měly. Respektive jak zobrazit webovou stránku pro zařízení s velkými i malými úhlopříčkami.

Responzivní web design je metoda, která tento problém řeší a design webových stránek učiní flexibilním, tudíž nemá pevnou velikost, ale na základě detekce velikosti obrazovky přizpůsobí design (obsah) tak, aby byl optimalizován pro danou obrazovku zařízení, ve kterém je webová stránka zobrazena. O responzivním webu jako první psal Ethan Marcotte v roce 2010. (Peterson, 2014) W3C v roce 2005 navrhlo standard CSS3, který již obsahoval *media query*, díky čemuž se na základě definované šířky obrazovky daly použít různé kaskádové styly na danou stránku, které přizpůsobily obsah stránky danému zařízení, aniž by změnily obsah samotný. (Rivoal, 2012, [online])

Důležitost responzivního designu zachycuje statistika z roku 2014, kdy 58 % Američanů vlastní chytrý telefon – telefon s operačním systémem iOS, Android, Windows Phone, které umožňují uživatelům plnohodnotný přístup k internetu. Dále 35 % Američanů vlastní tablet. Z toho důvodu se responzivní design stává standardem při tvorbě webu. (Peterson, 2014)

To, jak funguje responzivní design na čtyřech typech zařízení – monitor, notebook, tablet a chytrý telefon – lze lépe pochopit z grafického vyobrazení na obrázku č. Obr. 1 níže. Na každém ze zařízení je obsah v jiném rozložení, a tudíž i lépe čitelný pro danou obrazovku. V případě široké obrazovky je obsah plnohodnotně vyobrazen ve čtyřech sloupcích, ale se zmenšující se šířkou se obsah zužuje a sloupce v řádku ubývají a přesunují se na další řádek. U nejužších obrazovek je nakonec obsah uzpůsoben často jednosloupcově. (Tutorial Republic, [online])

Obr. 1 Ukázka responzivního designu (Tutorial Republic, 2016, [online])

V posledních letech se v moderním webovém designu webových aplikací využívá *flat a material design*, který je založen na tom, že se design co nejvíce zjednodušuje, používají se jednolité barvy, odstraňují se nepodstatné prvky, mezi něž patří například stínování a zkosené tvary, a je kladen důraz na typografické zpracování a použitelnost. Flat a material design vychází z minimalistické jednoduchosti, kdy se šetří grafickými prvky a barvami, mnohdy se taktéž dělají jednostránkové weby. Flat a material design je tudíž ideální pro responzivní weby a především pro mobilní zařízení.

Jeho nevýhodou ovšem je, že jej nelze použít všude, respektive ne na všechny obory. Příkladem může být státní správa, weby obcí a celkově weby veřejných i soukromých institucí, kde se nachází velké množství informací, které nelze příliš zjednodušovat, a tudíž by se porušila minimalistická jednoduchost, na které tento design zakládá. Také by to mohlo způsobit, že by se uživatel nemusel dostatečně orientovat na daném webu.

V poslední době se ustupuje od obrázků, které zobrazují informace na webu – například tlačítko s obrázkem domečku pro přechod na domovskou stránku. Web designéři využívají v mnoha případech raději sadu ikon, které se dají použít jako font symbolizující danou informaci. Jedná se především o sadu ikon FontAwesome, která obsahuje speciální vektorové ikony. Ty lze různě zvětšovat, aniž by ztrácely na kvalitě, tudíž jsou ideální pro responzivní design a mobilní zařízení, kdy není potřeba komprimovat obrázek, ale pouze použít font, vyobra-

zující požadovanou akci, čímž se šetří i objem přenášených dat a tím i rychlost načítání webu.

3.2 Bootstrap

Od roku 2013 se stal Bootstrap framework jedním z nejvíce populárních projektů ve verzovacím nástroji Git. To vše i díky dobré podpoře komunity a širokému ekosystému obsahujícím šablony a rozšíření kolem něj. Bootstrap ušetří programátorovi spoustu času tím, že zjednodušuje samotné psaní kódu, čímž dovoluje se zaměřit na samotnou tvorbu webu. Bootstrap byl původně vytvořen jako framework podporující konzistenci mezi interními nástroji společnosti Twitter a od verze 3 je licencován pod open source licencí MIT.

Mezi silné stránky tohoto frameworku patří responzivní mřížka, neboli tabulkový systém, který se přizpůsobí velikosti obrazovky. Tento tabulkový systém využívá předdefinovaných CSS tříd, které definují řádky a sloupce. Třída *row* slouží pro vyváření horizontálních skupin – řádků. V těchto řádcích se pracuje se sloupci, jejichž celkový počet může nabývat maximální hodnoty 12, do nichž se vkládá obsah. Tyto sloupce využívají třídu, která je ve tvaru *col-rozlišení-počet_sloupců*. Rozlišení těchto tříd je čtyř typů, které mají zkratku na základě rozlišení základních zobrazovacích zařízení uvedených na obrázku č. 6.

- *col-xs* pro extra malá (extra small) zařízení s rozlišením < 768 px (chytré telefony)
- *col-sm* pro malá (small) zařízení s rozlišením ≥ 768 px (tablety)
- *col-md* pro středně velká (medium) zařízení s rozlišením ≥ 992 px (notebooky)
- *col-lg* pro velká (large) zařízení s rozlišením ≥ 1200 px (monitory)

Počet_sloupců definuje, kolik sloupců (bráno zleva) bude seskupeno dohromady. (Bootstrap, [online])

3.3 JavaScript

JavaScript je objektově orientovaný skriptovací jazyk zpracovávaný na straně klienta. Je využíván především pro interakci s webovými stránkami. K tomu využívá DOM (Document Object Model), který poskytuje metody a rozhraní pro práci s obsahem webové stránky. DOM umožňuje JavaScriptu přidávat, odebírat a upravovat HTML elementy a jejich atributy. Taktéž lze k různým elementům přiřadit události (events), které spouští předdefinované metody JavaScriptu. Událostí může být například kliknutí myši (`onClick`) na prvek stránky nebo změna prvku (`onChange`) v případě změny hodnoty v definovaném prvku na stránce. (Zakas, 2012)

3.4 jQuery

Knihovna jQuery, která je v současné době nejvíce používanou JavaScriptovou knihovnou, usnadňuje a především zrychluje vývoj webových stránek. Výhodou této knihovny je mnohem jednodušší syntaxe oproti čistému JavaScriptu, tudíž dříve složité úkoly lze pomocí jQuery udělat mnohem jednodušeji. Další výhodou je, že sama knihovna má malou velikost. Díky své rozšířenosti lze nalézt a použít již mnoho existujících řešení na konkrétní problémy. Dále má velmi kvalitní dokumentaci spolu s ukázkami praktických příkladů použití. (Cranley, 2011)

3.5 FormBuilder

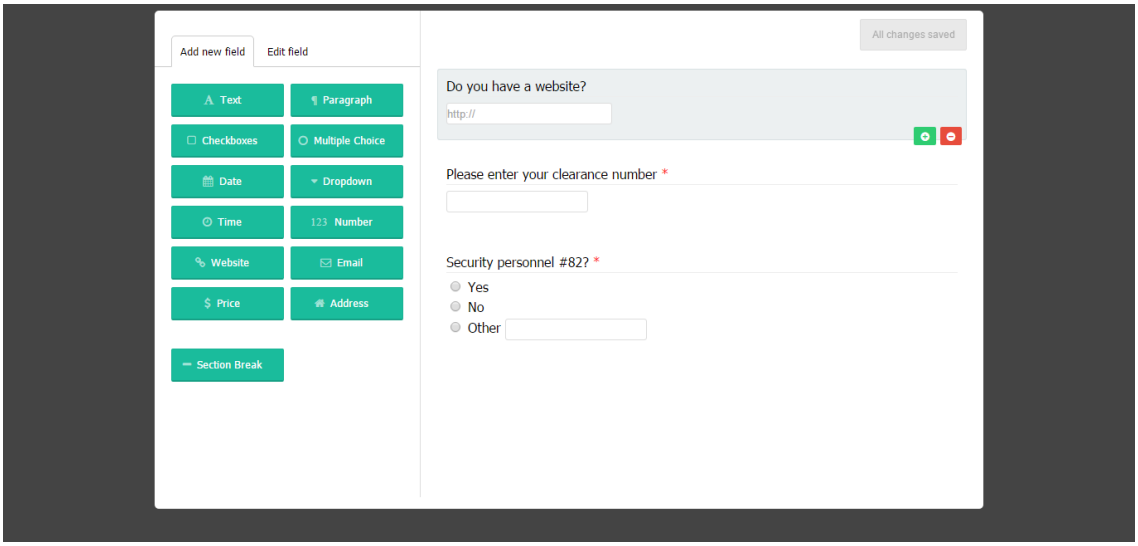
FormBuilder je již předpřipravené základní grafické rozhraní, které zvládá plnohodnotnou interaktivní tvorbu formulářů pomocí drag&drop. Kromě využití JavaScriptu a jQuery využívá ještě další externí knihovny, jako například Backbone & Rivets. FormBuilder je navržen tak, aby jej vývojář mohl do budoucna dále jednoduše rozšiřovat o další prvky. Jelikož se jedná pouze o grafické rozhraní, je samotné uložení formuláře taktéž ponecháno čistě na vývojáři. (GitHub, 2014, [online])

Samotné rozhraní FormBuilderu se skládá ze dvou částí. Pravá část slouží k zobrazení vytvořených prvků formuláře, levá část slouží k editaci těchto prvků a dalších vlastností formuláře. V této levé části FormBuilderu se nachází celkem dvě záložky:

- Add new field – slouží k vložení prvku do FormBuilderu
- Edit field – obsahuje vlastnosti vybraného prvku ve formuláři

Uživatel musí v prvním kroku vybrat prvek v záložce Add new field, který chce do formuláře vložit. Může to udělat dvěma způsoby – buď kliknutím na daný prvek, což jej přidá na poslední místo za ostatními prvky, nebo jeho přetažením do pravé části FormBuilderu na jakékoliv místo mezi již existující prvky. Jakmile vloží prvek do formuláře, automaticky se levé menu přepne na záložku Edit field, kde následně může upravovat vlastnosti prvků. Tyto vlastnosti jsou často závislé na tom, o jaký prvek se konkrétně jedná, neboť ne všechny prvky mají stejné vlastnosti.

Mezi společné vlastnosti prvků patří titulek, popisek a nastavení, zda se jedná o povinné či nepovinné pole. Tyto vlastnosti se okamžitě projevují v ukázce v pravé části FormBuilderu, takže uživatel ihned vidí, jak daný prvek vypadá. Následně může v závislosti na prvku upravovat jeho individuální vlastnosti, jako je jeho velikost, minimální/maximální počet znaků a podobně.



Obr. 2 Ukázka tvorby formuláře v základní verzi FormBuilderu

4 Metodika

Na začátku bude provedena analýza konkurenčních webových aplikací, na jejichž základě budou stanoveny specifikace na funkcionalitu formulářů, které pomohou vytvořit úvodní představu o samotné aplikaci. Na základě této analýzy bude vytvořen návrh celé webové aplikace. K tomuto účelu budou v programu Astah Professional vytvořeny diagramy případů užití, díky kterým bude znázorněna interakce uživatelů s aplikací. Následně bude proveden grafický návrh samotné webové aplikace a taktéž bude navržena vhodná struktura databáze pomocí ERD v programu PowerDesigner.

Při implementaci řešení bude použit PHP framework Nette a další vybrané technologie, jako je JavaScriptová knihovna jQuery. Aplikace bude zkoušena na lokálním serveru. Během implementace bude průběžně testována správná funkcionalita aplikace a budou se odlaďovat chyby. Následně bude aplikace nasazena na webhosting, kde bude testována uživateli. Na základě výsledků testování a připomínek uživatelů budou dále implementovány další funkce.

5 Analýza konkurenčních řešení

Webových aplikací, které se zabývají tvorbou formulářů, existuje na internetu nepřeberné množství. Od aplikací, které jsou zdarma, až po aplikace, za jejichž použití musí uživatel zaplatit buď jednorázově, nebo si platí pravidelné poplatky. Jejich kvalita, možnosti editace a další vlastnosti, jako je uživatelská přívětivost, intuitivnost a rychlost tvorby formulářů, taktéž možnosti zobrazování odpovědí a jejich vyhodnocení, se mnohdy diametrálně liší. V závislosti na verzi aplikace (zdarma nebo placená) lze zobrazovat další možnosti, jako jsou například statistiky – od základních až po podrobné. Proto v podkapitolách níže budou některé z nejznámějších nebo zajímavých konkurenčních aplikací analyzovány a získané informace budou použity pro následný návrh a vývoj.

Jelikož konkurenčních aplikací existuje velké množství, pro analýzu byly vybrány nejznámější a dle mého názoru i nejzajímavější aplikace. Bylo využito shrnutí 14 alternativ z blogu zappier.com (Matthew Guay, 2016, [online]) a makeuseof.com (Smith Brian, 2016, [online])

5.1 Google formuláře

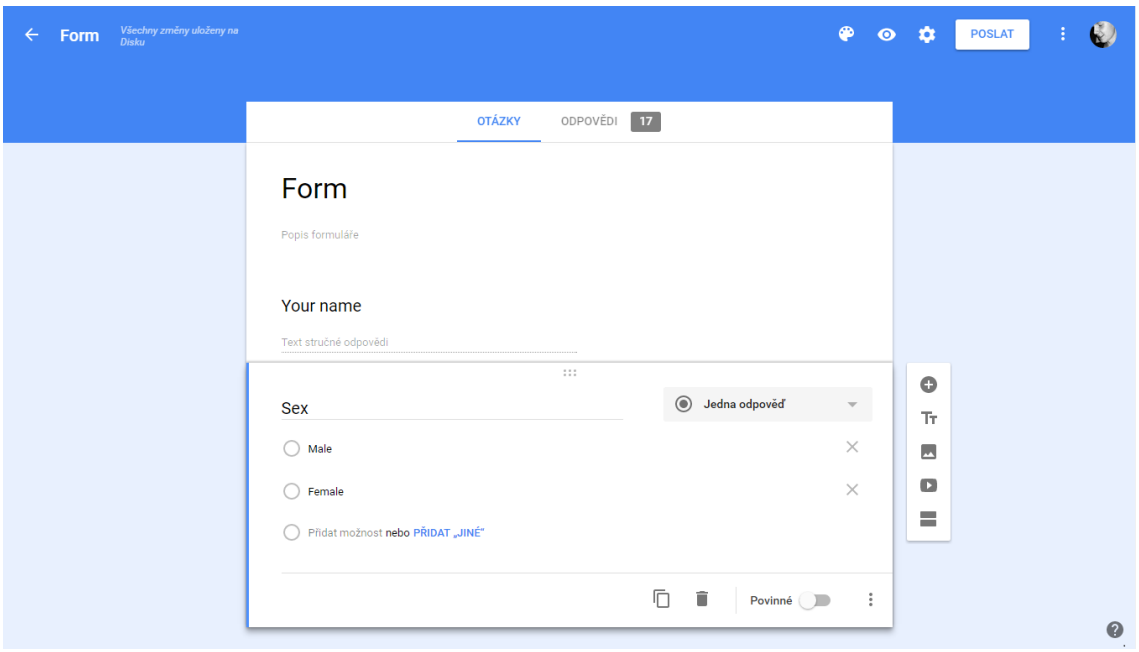
Google formuláře jsou pravděpodobně nejznámější aplikace pro tvorbu dotazníkových formulářů. Jsou dostupné zdarma na stránkách Google a jediné, co je potřeba, je mít u Google účet.

Aplikaci tvoří samotný formulář a vedle něj po pravé straně úzký plovoucí panel s možnostmi přidání prvků (Otázka, Název a popis, Obrázek, Video, Oddíl). Nejčastější používaný prvek je „Otázka“, který přidá implicitně jeden radio button, a ve většině případů je nutné ve vlastnostech prvku vybírat, jaký typ má být – například „Stručná odpověď“, „Rozbalovací nabídka“ a podobně. Z uživatelského hlediska je pro nastavení požadovaného prvku potřeba více kliků, a tudíž i více času, než uživatel nastaví vlastnosti prvku dle svých představ.

Vyhodnocení odpovědí probíhá ve stejném okně, kde si uživatel může pouhým překliknutím zobrazit souhrnné shrnutí odpovědí ze všech formulářů v podobě grafů a tabulek, případně jako individuální odpovědi, kde se mu zobrazí celý formulář s respondentovými odpověďmi. Nechybí zde taktéž možnost exportovat odpovědi přímo do excelovské tabulky (dostupné na docs.google.com/spreadsheet).

Mezi další možnosti formulářů v této aplikaci je náhodné pořadí otázek, zobrazování ukazatele postupu, tvorba kvízů, možnost změnit pozadí formuláře, případně vložit obrázek do hlavičky. Tyto možnosti jsou k nastavení v horní liště okna.

Největší nevýhodou zde shledávám to, že tato aplikace neposkytuje žádné statistiky. (Formuláře Google, 2016, [online])



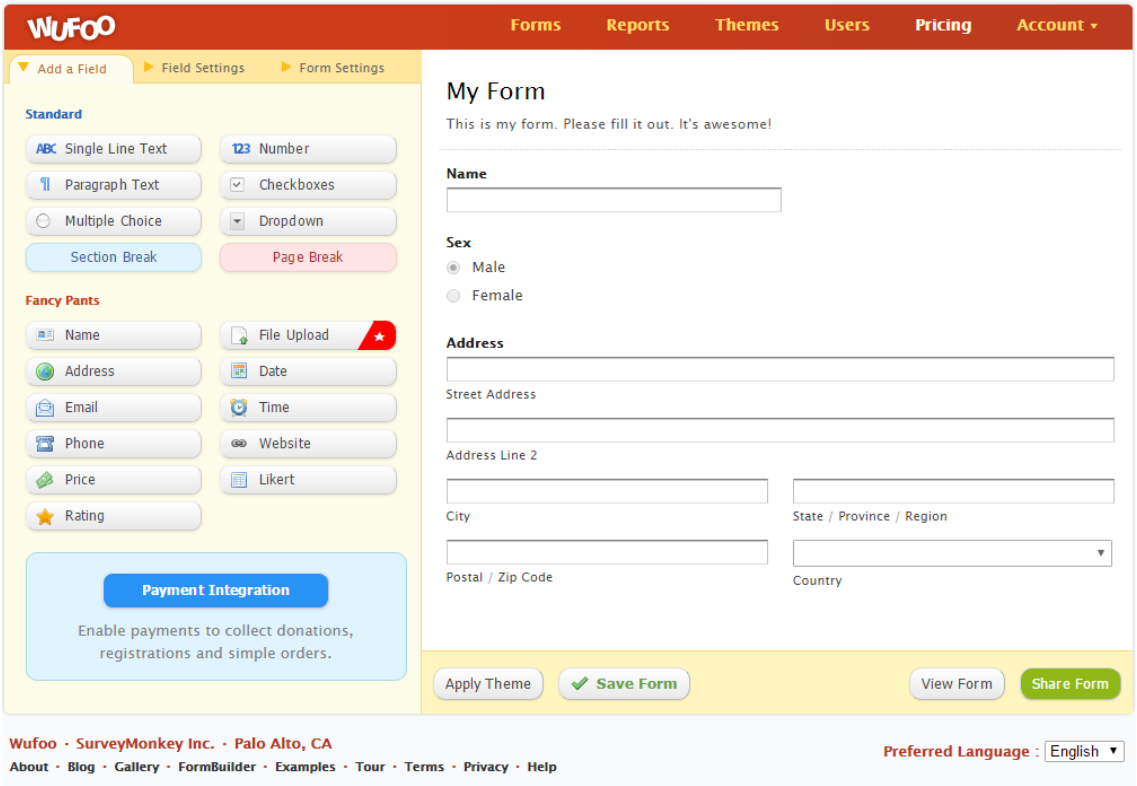
Obr. 3 Ukázka tvorby formuláře pomocí webové aplikace Google Forms

5.2 WuFoo

WuFoo aplikace je další z mnoha konkurenčních produktů, která je na trhu od roku 2006. Za tu dobu uživatelé vytvořili již milion formulářů. Aplikaci lze sice využívat ve variantě zdarma (GRATIS), která má omezení v počtu vytvářených formulářů, množství odpovědí na dané formuláře, polí a reportů, ale pokud uživatel potřebuje například získat více než 100 odpovědí od respondentů nebo dotazník obsahující více než 10 otázek, tak je bezplatná verze nedostačující. V případě větších nároků na formuláře by musel využít některou z placených variant v závislosti na tom, jaké má požadavky. Placené varianty jsou celkem tři. Nejlevnější verze AD HOC vyjde zaokrouhleně na 15 dolarů, vyšší verze BONA FIDE na 30 dolarů a nejdražší verze CARPE DIEM, která je neomezená, stojí 70 dolarů.

Co se týče samotného používání aplikace, její ovládání je velmi intuitivní a přehledné. Základní vložení prvků je velmi jednoduché, neboť jsou všechny prvky seřazeny v levém menu a rozpoznání jejich typu je na první pohled ihned patrné. Při použití drag&drop se hůře vkládá prvek na požadované místo a uživatel při přetahování prvku na volné místo musí přejet na již existující vložený prvek, aby se mu nabídla možnost jej vložit. Dalším nedostatkem shledávám to, že při vložení prvku se automaticky nenabízí možnost jeho editace a uživatel je tedy nucen buď na vložený prvek kliknout (vyvolá editační okno), nebo ručně kliknout v levém sloupci na záložku editace. Ne vždy aplikace reaguje správně – v mém případě například se u select boxu zobrazí editační okno až na několikátý pokus.

V aplikaci má uživatel dále možnost dělat vlastní statistiky, neboli reporty – uživatel si je vytváří sám dle svých potřeb. V těchto reportech si může vytvářet tabulky, grafy, zvolit si jejich zobrazení, zda vedle sebe, pod sebe, případně využít dalších možností zobrazení. Opět je zde horší ovládání pomocí drag&drop, navíc nelze jednoduše vytvořit report na celý formulář, ale uživatel musí ručně vybírat, na který prvek v dotazníku jej má pro daný report použít. Zde je patrná další nevýhoda verze zdarma, kdy si uživatel může vytvořit pouze tři reporty, tedy reporty na celkem tři prvky formuláře. (Wufoo, 2016, [online])



Obr. 4 Ukázka tvorby formuláře pomocí webové aplikace WuFoo

5.3 Typeform

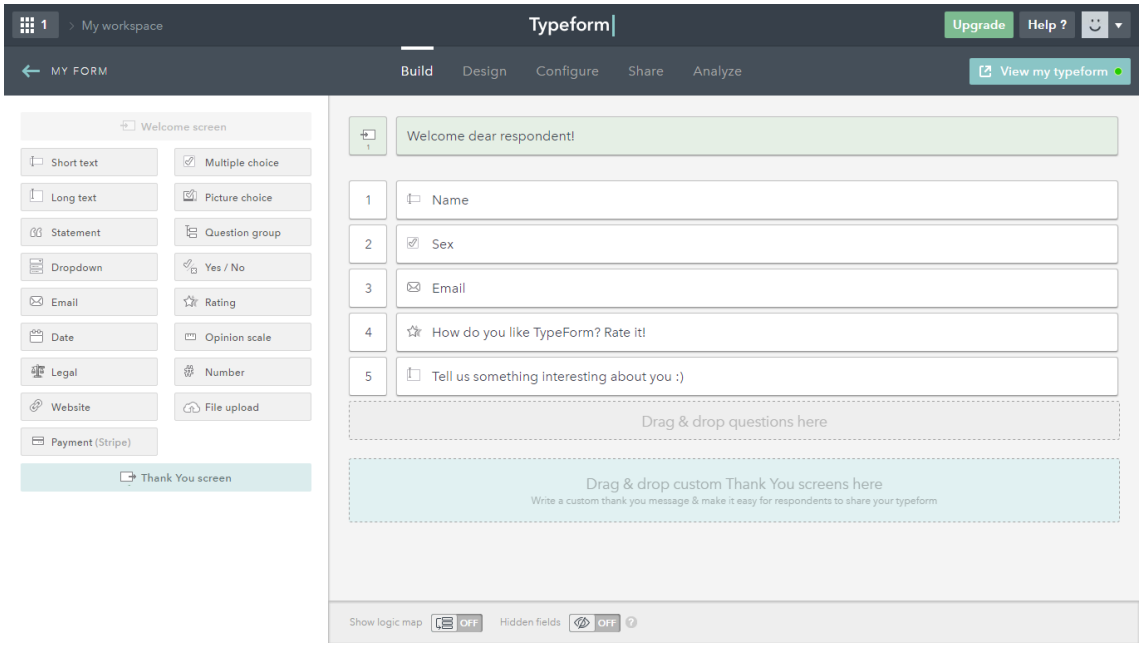
Aplikace TypeForm je na trhu od roku 2014 a nabízí tvorbu formulářů ve třech verzích – BASIC, který je zdarma, PRO (30 euro) a PRO+ (70 euro). Placené verze se liší nadstandardními možnostmi, které obyčejný uživatel pro své soukromé formuláře pravděpodobně nevyužije, ale například společnosti zabývající se prodejem je využijí – například prvek na platební údaje.

Slogan TypeFormu zní „Goodbye forms. Hello typeforms.“, což dává smysl při pohledu na vytvořený formulář, který je především interaktivní a nemá typický vzhled dotazníkových formulářů. Každý prvek formuláře je zde zobrazen

přes celou obrazovku a k další otázce se uživatel dostane kliknutím na tlačítko další, případně stiskem klávesy Enter.

Samotná tvorba formuláře je relativně jednoduchá, stejně jako u aplikace WuFoo se v levém menu nachází přehledný seznam prvků. Jednotlivé prvky obsahují spoustu nastavení, které na první pohled občas matou. Díky svojí nezvyklosti nelze při tvorbě formuláře okamžitě vidět v daném okně jeho výsledný vzhled, ale každý prvek je reprezentován jedním řádkem obsahujícím pouze jeho nadpis. Pokud si tedy uživatel chce zobrazit výsledný formulář, musí si jej zobrazit kliknutím na tlačítko „View my typeform“, což ale nese za následek to, že se již do statistik formuláře zaznamenává přístup, a tudíž dochází ke zkreslení statistik.

Velkým pozitivem této aplikace jsou podrobné statistiky. Metrická obsahuje informace o přístupu k formulářům podle zařízení – počítač, tablet, mobil, ostatní. U každého zařízení je uvedeno, kolik bylo k formuláři přístupů, počet odpovědí, průměrný čas potřebný k vyplnění formuláře a procento jeho dokončení. Dále je zde možnost propojit formulář s Google Analytics a tím pádem získat ještě podrobnější statistiky. (Typeform, 2016, [online])



Obr. 5 Ukázka tvorby formuláře pomocí webové aplikace TypeForm

5.4 Formstack

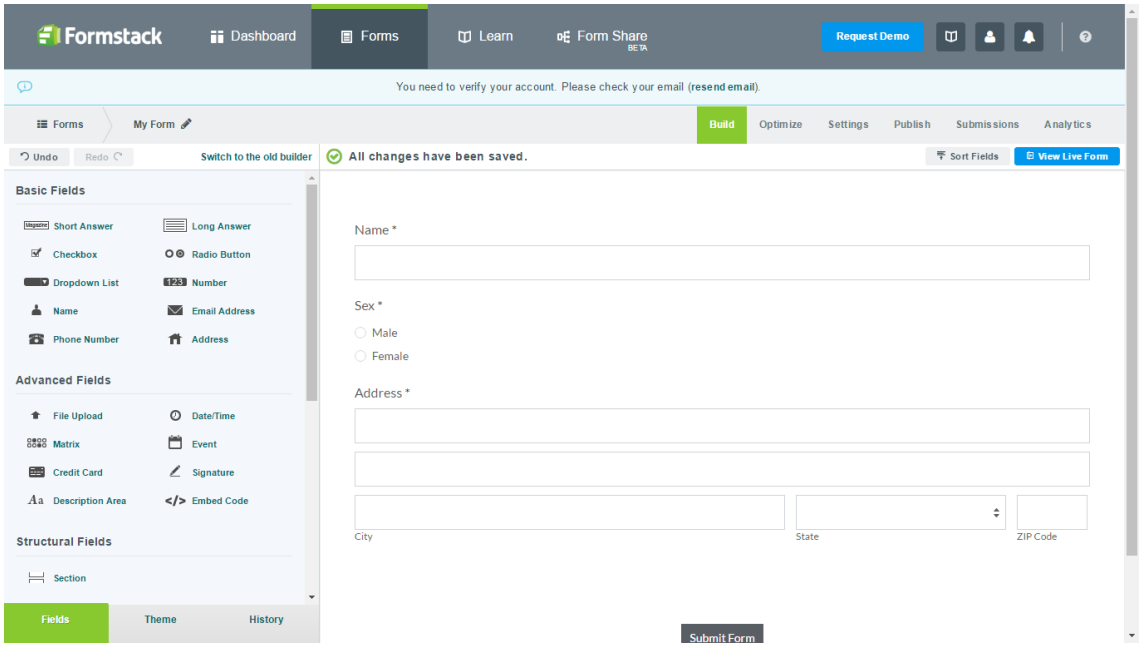
Další konkurenční aplikací na tvorbu formulářů je aplikace Formstack, která je dostupná pouze v placených verzích. Po čtrnáctidenní trial verzi, která je samozřejmě zdarma, má uživatel možnost si zakoupit některou ze tří placených verzí, a to buď nejlevnější Silver za 39 dolarů, která obsahuje pouze základní možnos-

ti, nebo vyšší verze Gold a Platinum (99 a 249 dolarů). Tyto verze již umožňují používat Google Analytics pro vedení podrobných statistik, propojení s Facebookem a podobně.

Samotná aplikace Formstack je cílena především na firmy a jejich marketing, kdy plně reflektuje jejich požadavky na funkcionalitu, například A/B testování, pokročilé analýzy výsledků a podobně. Pro běžného uživatele vhodná příliš není, neboť zde chybí alespoň základní verze, která by byla zdarma (nebereme-li v potaz to, že uživatel může využít čtrnáctidenní trial verzi).

Zobrazení odpovědí respondentů je velmi přehledné – uživatel má na výběr buď zobrazit odpovědi v tabulce, nebo pomocí grafů (v případě, že se jedná o odpovědi, které lze graficky vyhodnotit). Pro základní statistiky lze využít integrované metrické statistiky, kdy si uživatel může zobrazit za určité období počet respondentů na formulář, jejich konverzi, procentuální vyjádření konverze atp.

Nevýhodou zde lze shledat to, že pokud chce uživatel využít hlubší analýzy odpovědí, je potřeba používat nástroje třetích stran. To se týče například integrace HubSpot nástroje pro předání dat přímo do marketingového nástroje HubSpot, nebo integrace nástroje Salesforce, pro přímé ukládání dat do cloudové databáze Salesforce. Jelikož je aplikace Formstack určena především pro marketingové účely, lze očekávat, že mnoho oddělení tyto nástroje třetích stran více či méně využívá, tudíž je pro ně často Formstack vhodnou variantou. (Formstack, 2016, [online])



Obr. 6 Ukázka tvorby formuláře pomocí webové aplikace Formstack

5.5 JotForm

Asi nezajímavější a nejlépe zpracované formuláře, jak z hlediska jednoduchosti a přehlednosti, tak i z hlediska možností jejich nastavení, jsou dle mého názoru od společnosti JotForm. Tato společnost je stejně jako WuFoo na trhu od roku 2006 a v současné době využívá jejich služeb více než 2 miliony uživatelů. Nabízí se ve čtyřech verzích, kdy nejnižší verze (Starter) je zdarma a další (Bronze, Silver a Gold) jsou placené (19, 39 a 99 dolarů). Kromě množství odpovědí a formulářů je u placených verzí možnost propojit formuláře s platebními bránami a přidávání nadstandardních prvků.

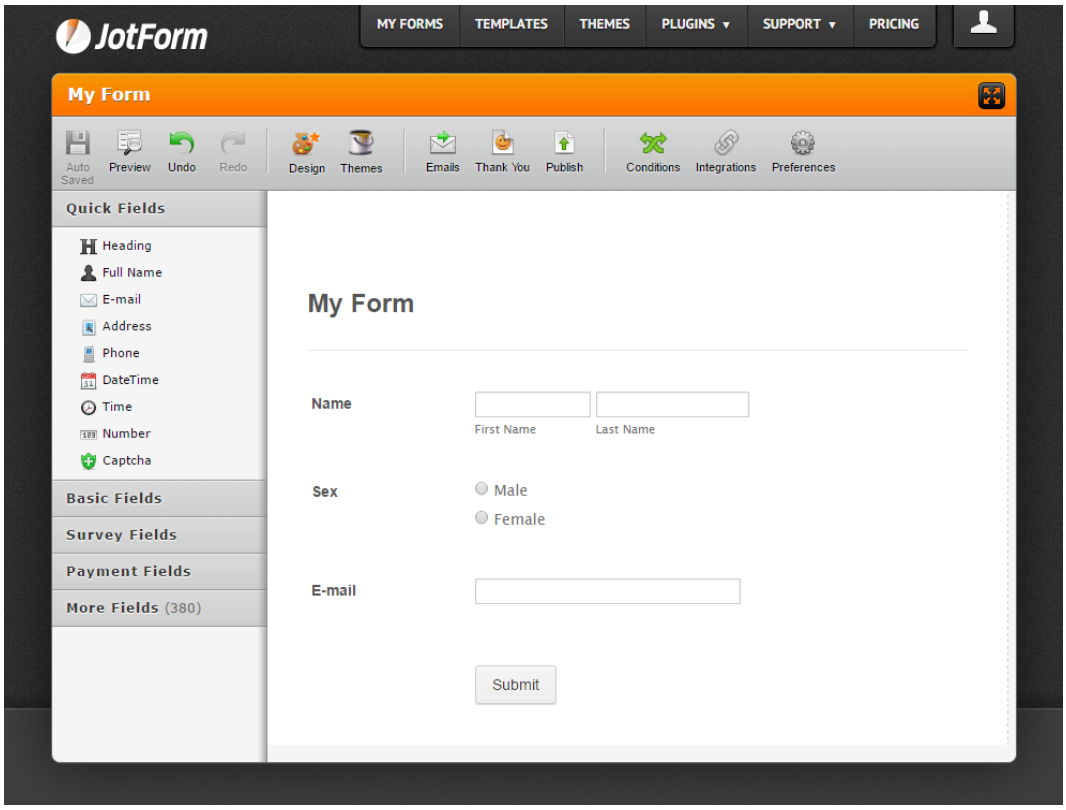
Samotná tvorba formuláře je jednoduchá. Uživatel si může vybrat z již předpřipravených formulářů, případně si může v editačním okně pomocí drag&drop během chvilky vytvořit vlastní formulář. Základní prvky v levém menu Quick Fields obsahují předpřipravené prvky jako je jméno, email, adresa, telefon a podobně. Pokud má uživatel individuální požadavky na vlastní prvky, lze je v Basic Fields vložit a připsat k nim popisky a dále upravit.

Krom toho má uživatel možnost vložit i hodnotící prvky jako je například hodnocení pomocí hvězd, matice. Mezi další prvky lze zmínit ještě zlom stránky, který rozdělí formulář na další stránku. Pokud tyto základní prvky stále nestačí, je k dispozici ve vlastním obchodě JotForm dalších 380 prvků – widgetů. Mezi nejpopulárnější widgety patří vkládání videa z YouTube, tvorba fotografie z webkamery respondenta, SMS potvrzení respondentova čísla, autovyplnění adresy pomocí kliknutí respondenta do mapy a podobně. Nevýhodou widgetů je, že ne všechny jsou responzivní, a tudíž mohou kazit celkový dojem z formuláře.

Následně může uživatel svému formuláři nastavit předpřipravené témata, která nastaví formuláři předdefinovaný vzhled, kdy kromě pozadí stránky a formuláře mohou ovlivnit i vzhled jeho prvků. Tato témata jsou opět k dispozici v obchodě JotForm, a to buď zdarma, nebo za jednorázový poplatek.

Co se týče odpovědí respondentů na formuláře, shledávám zde oproti výše uvedeným alternativám nedostatky v jejich prezentaci. Odpovědi jsou zobrazovány jako jednotlivé odpovědi respondentů na formulář, a to pouze v textové podobě. Chybí zde často využívaná prezentace odpovědí všech respondentů na jednotlivé prvky ve formuláři, například ve formě přehledných grafů.

Statistiky o respondentech jsou, stejně jako u aplikace TypeForm, metrické, ale na rozdíl od TypeForm nebo Formstack jsou méně přehledné. (JotForm, 2016, [online])



Obr. 7 Ukázka tvorby formuláře pomocí webové aplikace JotForm

5.6 Vyhodnocení analýzy konkurenčních formulářů

Z analýzy výše uvedených formulářů vyplynuly jejich přednosti a nedostatky. Některé nedostatky jsou dány tím, že dané formuláře je buď nemají implementované, nebo je mají implementované, ale uživatelsky nepřívětivě.

Jelikož byly analyzovány aplikace, které jsou nejznámější, nelze říci, že by jejich ovládání v případě tvorby formuláře bylo nevyhovující. Díky široké uživatelské základně je jejich rozhraní, až na některé výše uvedené nedostatky, dostatečně vyladěné a uživatelsky přívětivé.

Naopak nedostatky se vyskytují v další životní etapě, kdy se vytvořený formulář rozešle respondentům. To se týče především statistik vedených o respondentech, které jsou kromě TypeForm nepřehledné, případně úplně chybí. Pro mnoho uživatelů je důležité vědět, jaká je úspěšnost vyplnění dotazníků, kolik času průměrně respondent stráví nad dotazníkem, v jaké části přestane dotazník vyplňovat, případně z které části světa respondent pochází.

Dalším, možná největším nedostatkem těchto aplikací shledávám vyhodnocování odpovědí respondentů, kdy často chybí jednoduché a přehledné vyhodnocení. Aplikace Google formuláře a Formstack ze všech výše analyzovaných aplikací mají toto vyhodnocení nejlépe zpracované, především graficky. WuFoo sice vyhodnocení má, ale nutí uživatele ručně zadávat, které prvky a jakým způ-

sobem chce vyhodnocovat, taktéž samotná tvorba reportů je nepřívětivá a omezená.

Při tvorbě formulářů je dále častým nedostatkem to, že uživatel ve většině případů nevidí ihned na první pohled, jak vytvářený formulář vypadá. Samozřejmě si může zobrazit náhled, ale to je již o další krok pro uživatele navíc. Nejlépe v tomto ohledu dopadly formuláře od Google, kde lze vidět okamžitě výsledný formulář již při jeho tvorbě.

5.7 Specifikace na funkcionalitu formulářů

Z dané analýzy a následného vyhodnocení vyplývá, že by vytvořená aplikace měla mít (v některých případech alespoň na základní úrovni) následující vlastnosti:

Jednoduchost tvorby formuláře – uživatel by měl během krátké chvíle zvládnout vytvořit formulář. Přidání prvku bude možné dvěma způsoby – kliknutím nebo přetáhnutím prvku pomocí drag&drop. Po jeho přidání bude mít uživatel okamžitě k dispozici editační menu, kde si uživatel nově přidaný prvek bude moci editovat dle svých požadavků.

Formulář bude již při jeho tvorbě okamžitě zobrazen tak, jak bude vypadat před respondentem, který jej bude vyplňovat.

Odpovědi na formuláře budou vyhodnoceny dvěma základními způsoby – tabulkově a graficky. Zároveň si uživatel bude moct zobrazit odpovědi dvěma různými způsoby – jednotlivé odpovědi na formuláře, hromadné odpovědi na konkrétní otázku (resp. prvek formuláře).

V případě grafického vyhodnocení odpovědí si uživatel bude moci vybrat z několika typů grafů tak, aby byly plně reflektovány uživatelovy požadavky na výstup.

Odpovědi si uživatel bude moci stáhnout ve formátu CSV (comma-separated values, hodnoty oddělené čárkami) pro další využití v softwaru třetích stran.

Aplikace bude poskytovat přehledné zobrazení statistik, alespoň základní metrické, aby uživatel měl přehled o chování respondentů.

Kromě samotné funkcionality tvorby formulářů bude aplikace splňovat požadavky týkající se zabezpečení – aplikace by měla zvládat správu uživatelských rolí a umožňovat uživatelům přístup do částí aplikace podle jejich oprávnění.

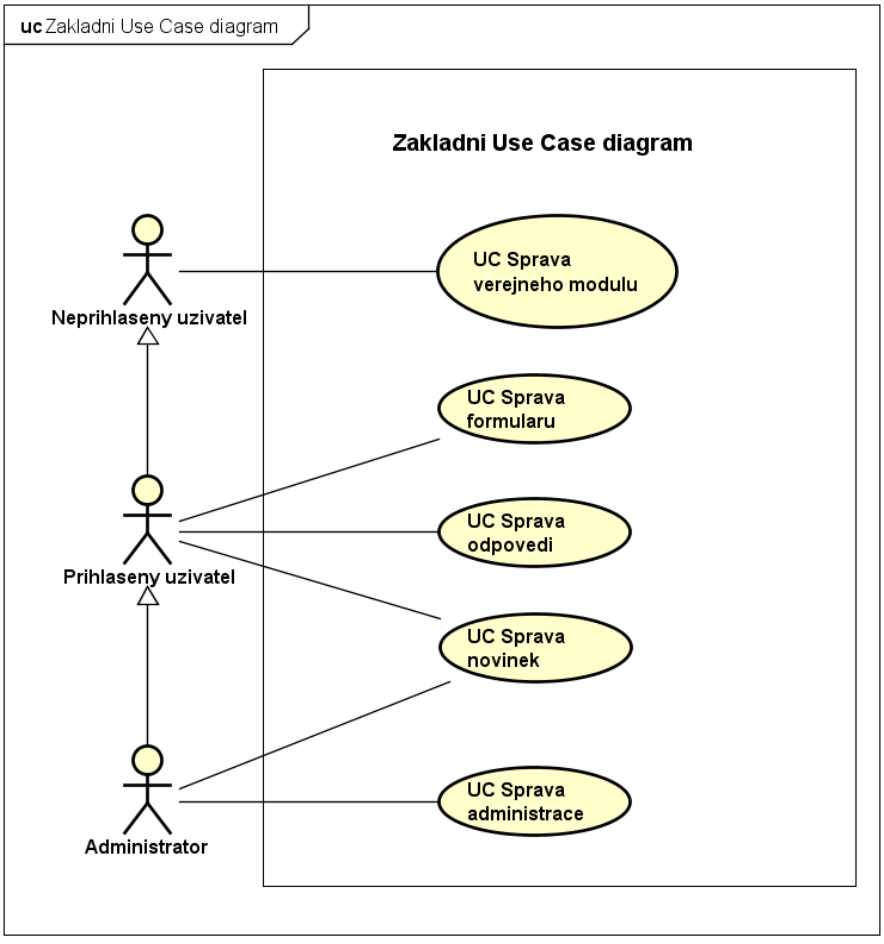
Implementace proběhne v PHP jazyce s využitím frameworku Nette, jehož součástí je automatické escapování vstupních hodnot od uživatele, ochrana proti Cross-Site scripting (XSS) a SQL inject útokům. Taktéž bude přizpůsobena jak pro zobrazení na monitorech, tak i na zařízeních s menším rozlišením, jako jsou mobilní telefony a tablety. K tomu bude využito frameworku Bootstrap 3.

6 Návrh aplikace

Pro popis interakce mezi aplikací a uživateli jsou použity diagramy případu užití. Základní diagram zobrazuje uživatelské role a k nim patřící základní případy užití, které jsou z důvodu přehlednosti následně dekomponovány a blíže popsány.

6.1 Uživatelské role

Z obrázku níže je patrné, že aplikaci by měly využívat celkem tři typy uživatelů. Nepřihlášený uživatel má přístup pouze do veřejné části aplikace. Přihlášený uživatel má na rozdíl od nepřihlášeného uživatele přístup i ke správě formulářů, správě odpovědí a taktéž i do správy novinek. Od přihlášeného uživatele dědí stejná práva administrátor, který má navíc přístup do správy administrace.

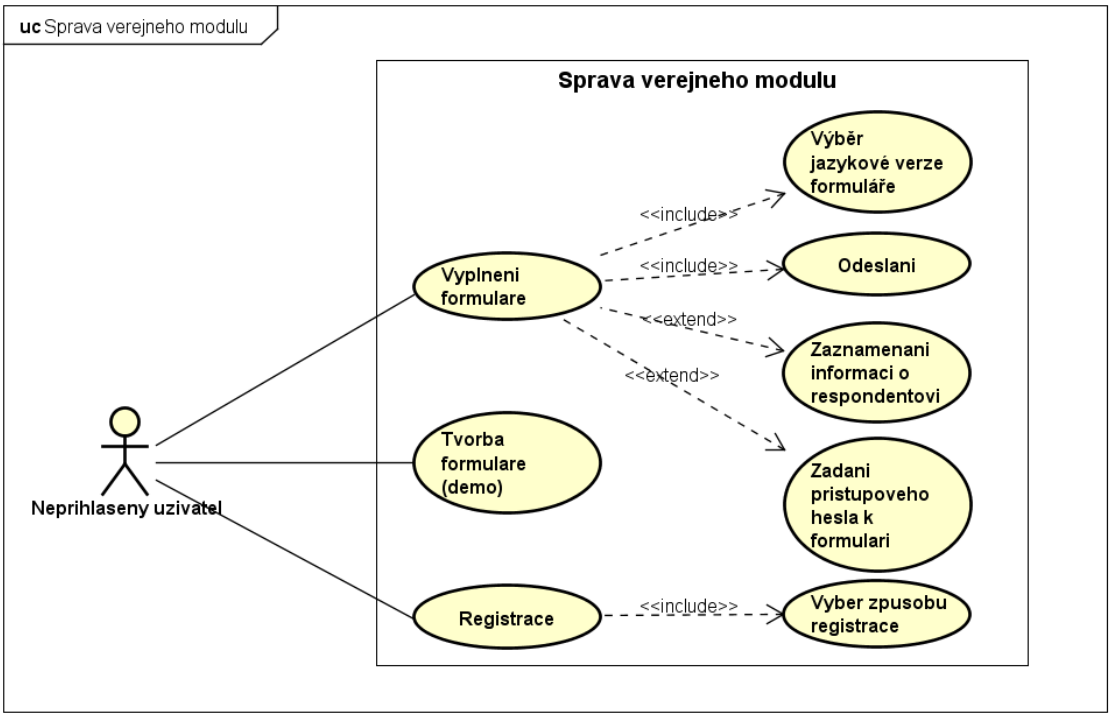


Obr. 8 Případy užití základní rozvržení

6.2 Správa veřejného modulu

První dekomponovaný případ užití se zabývá vztahem, který popisuje, co může v aplikaci provádět nepřihlášený uživatel. Nepřihlášený uživatel, ve smyslu respondenta, má možnost vyplnit formulář a následně jej odeslat. Ve chvíli vyplňování formuláře je zaznamenána informace o respondentovi pro budoucí statistiky. V případě, že je formulář chráněn heslem, je po tomto uživateli před samotným odpovídáním na formulář požadováno toto heslo zadat.

Dále má možnost registrace, aby mohl využívat dalších možností aplikace. Při registraci si navíc může vybrat z několika způsobů. Zároveň má možnost si v tomto veřejném modulu vyzkoušet tvorbu formuláře, avšak vytvořený formulář nebude moci uložit.



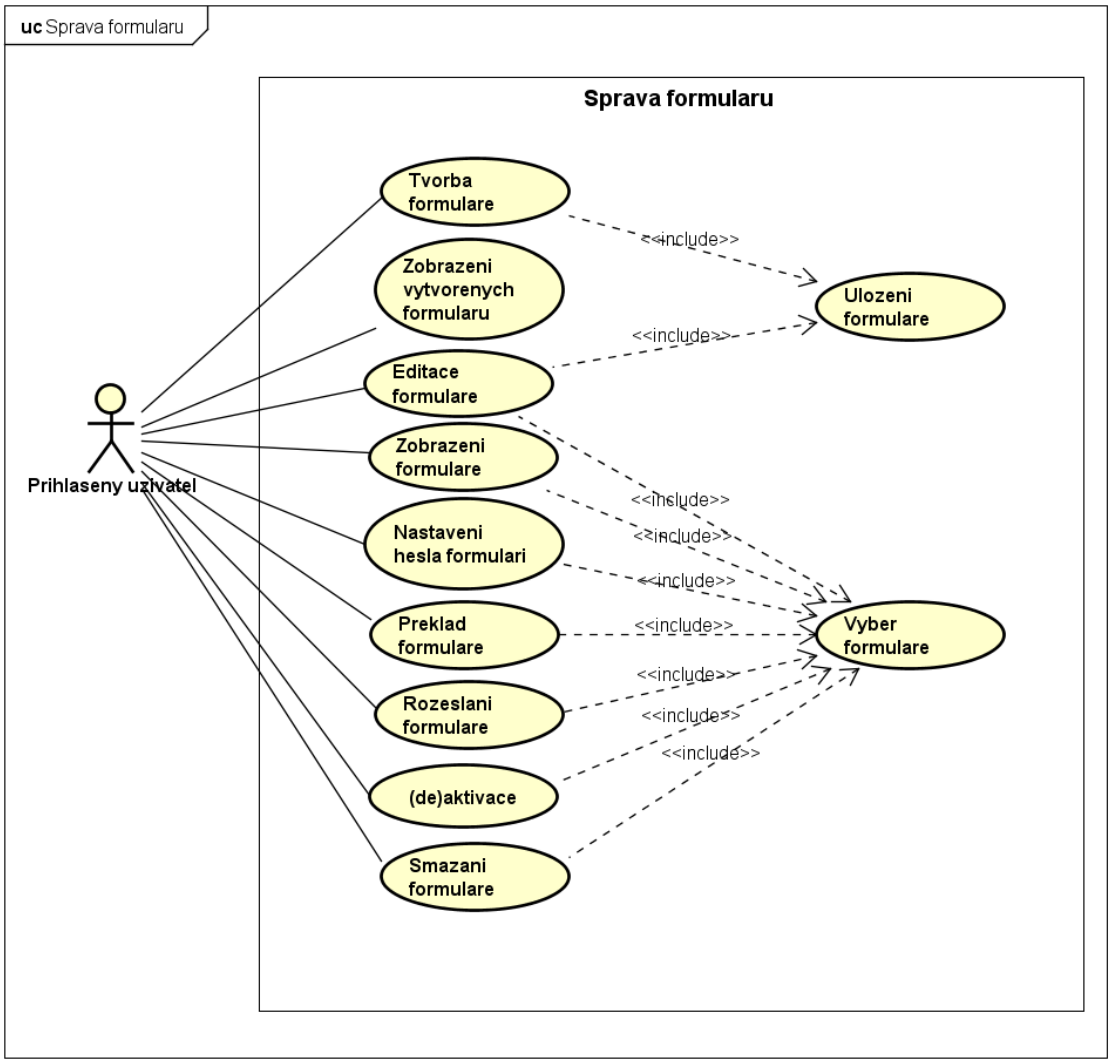
Obr. 9 Případy užití nepřihlášeného uživatele

6.3 Správa formulářů

Přihlášený uživatel má oproti nepřihlášenému uživateli možnost vytvořit formulář a ten následně uložit. Vytvoří-li si více formulářů, může si je všechny přehledně zobrazit. Takto vytvořené formuláře mohou zobrazit, případně dále editovat. Formulářům lze nastavit heslo pro případ, že by chtěli, aby na něj mohli odpovídat pouze lidé, kterým bylo toto heslo poskytnuto. Kromě toho lze vytvořené formuláře překládat do více jazyků.

Vybrané formuláře lze rozesílat respondentům, v případě ukončení dotazování lze konkrétní formulář deaktivovat, aby na něj již nebylo možné odpovídat.

Samozřejmostí je i to, že uživatel má možnost jakýkoliv ze svých formulářů kdykoliv zcela smazat.



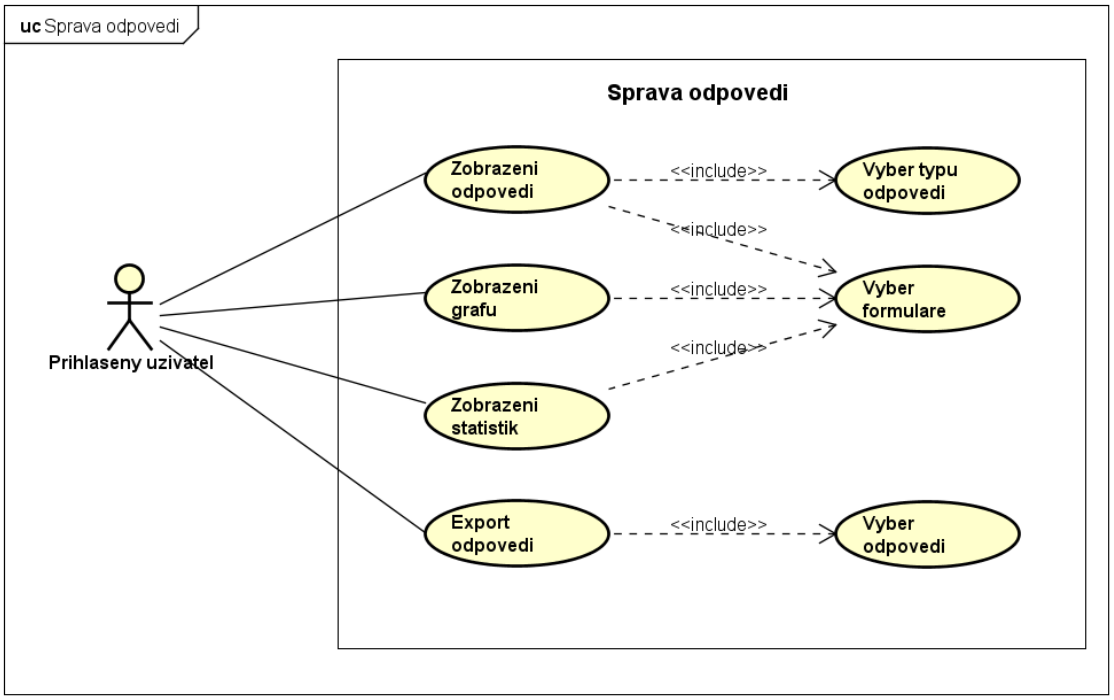
Obr. 10 Případy užití správy formulářů

6.4 Správa odpovědí

Jestliže na formulář začnou odpovídat respondenti, respektive nepřihlášení uživatelé z obrázku Obr. 9, má vlastník formuláře možnost si tyto odpovědi zobrazit, a to na základě výběru z několika typů zobrazení odpovědí.

Tyto odpovědi si mohou zobrazit taktéž v grafické podobě za pomocí grafů a zároveň mohou nahlížet do statistik, díky kterým zjistí bližší informace o tom, jak respondenti odpovídali na daný formulář.

Nechybí zde ani možnost tyto odpovědi exportovat, případně vybrat odpovědi, které mají být exportovány.

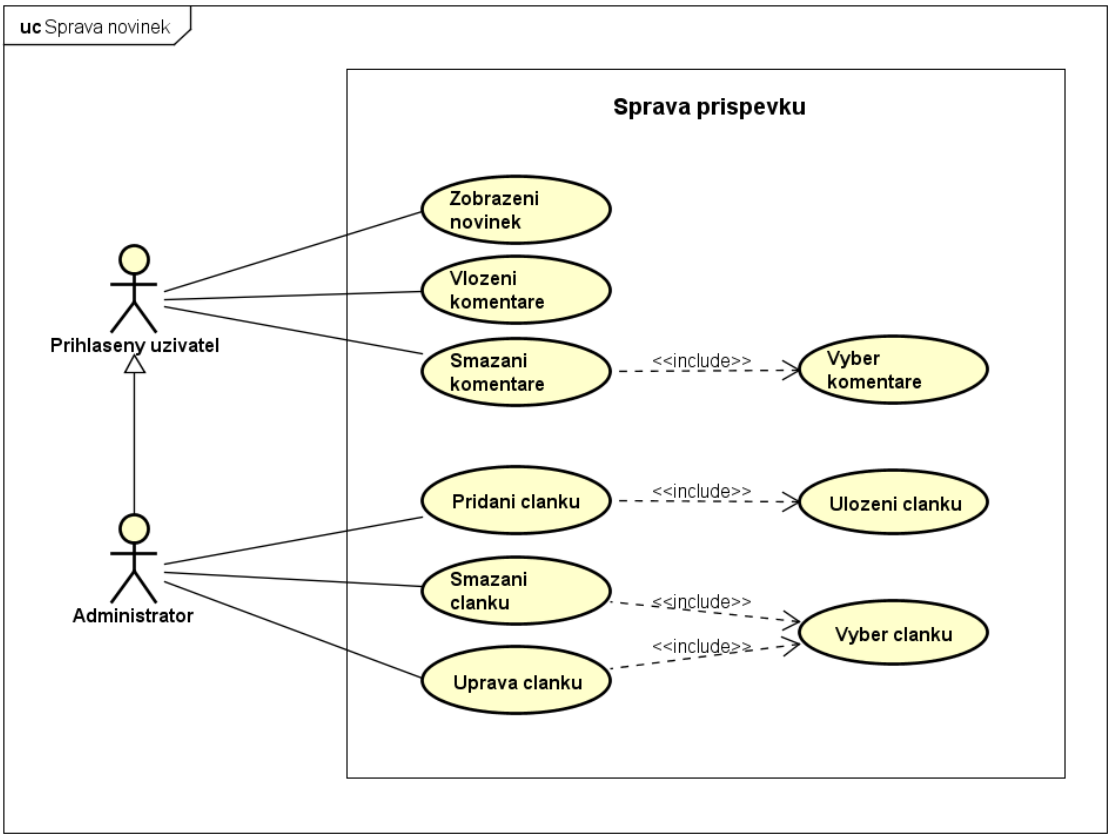


Obr. 11 Případy užití správy odpovědí

6.5 Správa příspěvků

Přihlášený uživatel si kromě správy formulářů a jejich odpovědí může zobrazit v systému příspěvky v sekci Novinky. Tyto příspěvky má možnost komentovat. Svoje komentáře může případně kdykoliv smazat.

Od Přihlášeného uživatele dědí jeho vlastnosti Administrátor, který může přidávat a ukládat nové články, případně tyto články mazat nebo upravovat.

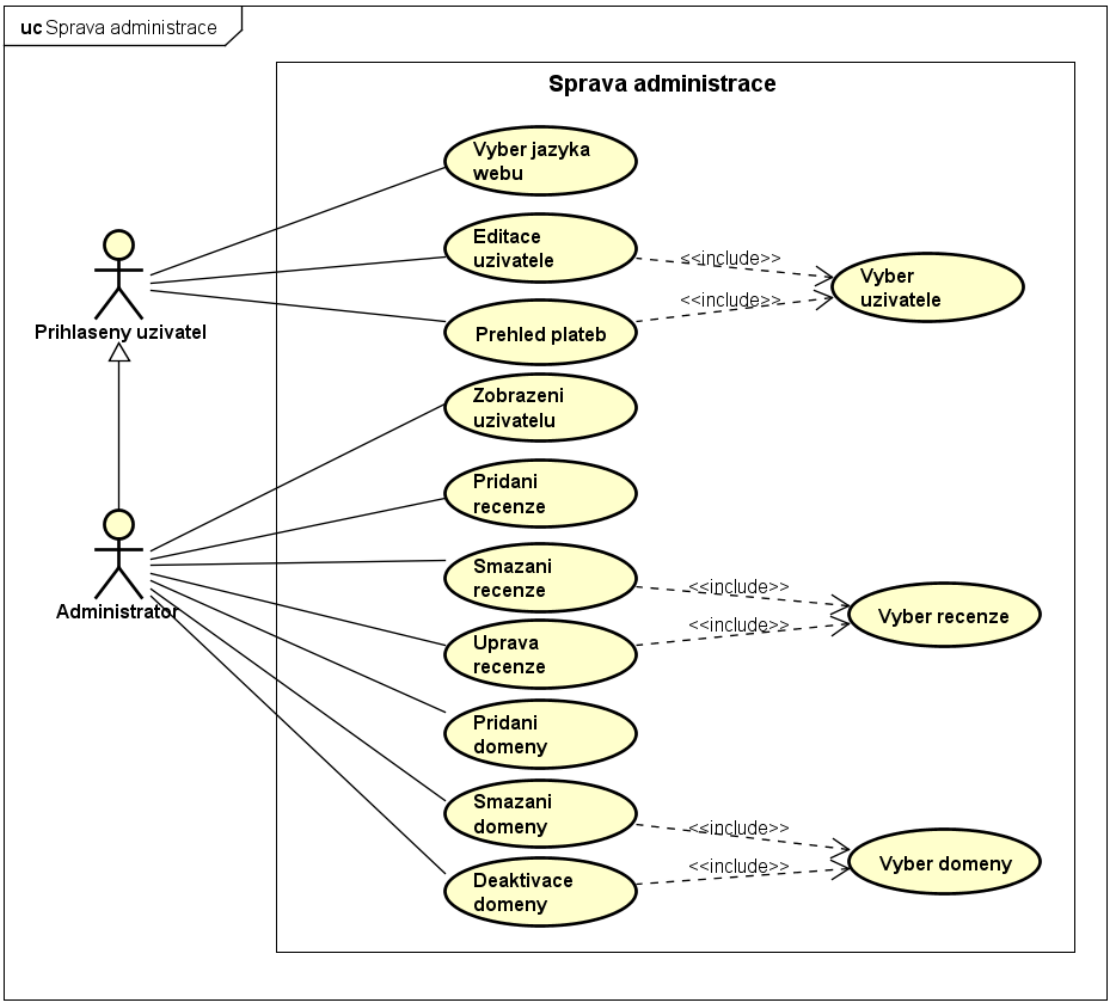


Obr. 12 Případy užití správy příspěvků

6.6 Správa administrace

Případ užití Správa administrace obsahuje dva aktéry. Prvním z nich je přihlášený uživatel, který má možnost vybrat jazyk, ve kterém má být web zobrazován, dále může upravovat svůj osobní profil a získat přehled svých plateb za využívání služeb.

Administrátor dědí od přihlášeného uživatele všechny výše zmíněné případy užití, navíc může zobrazit všechny registrované uživatele a ty dále upravovat. Kromě přidávání příspěvků do novinek může přidávat, upravovat a mazat recenze od spokojených zákazníků a taktéž provádět administrativu domén, ze kterých lze k dané aplikaci přistupovat.



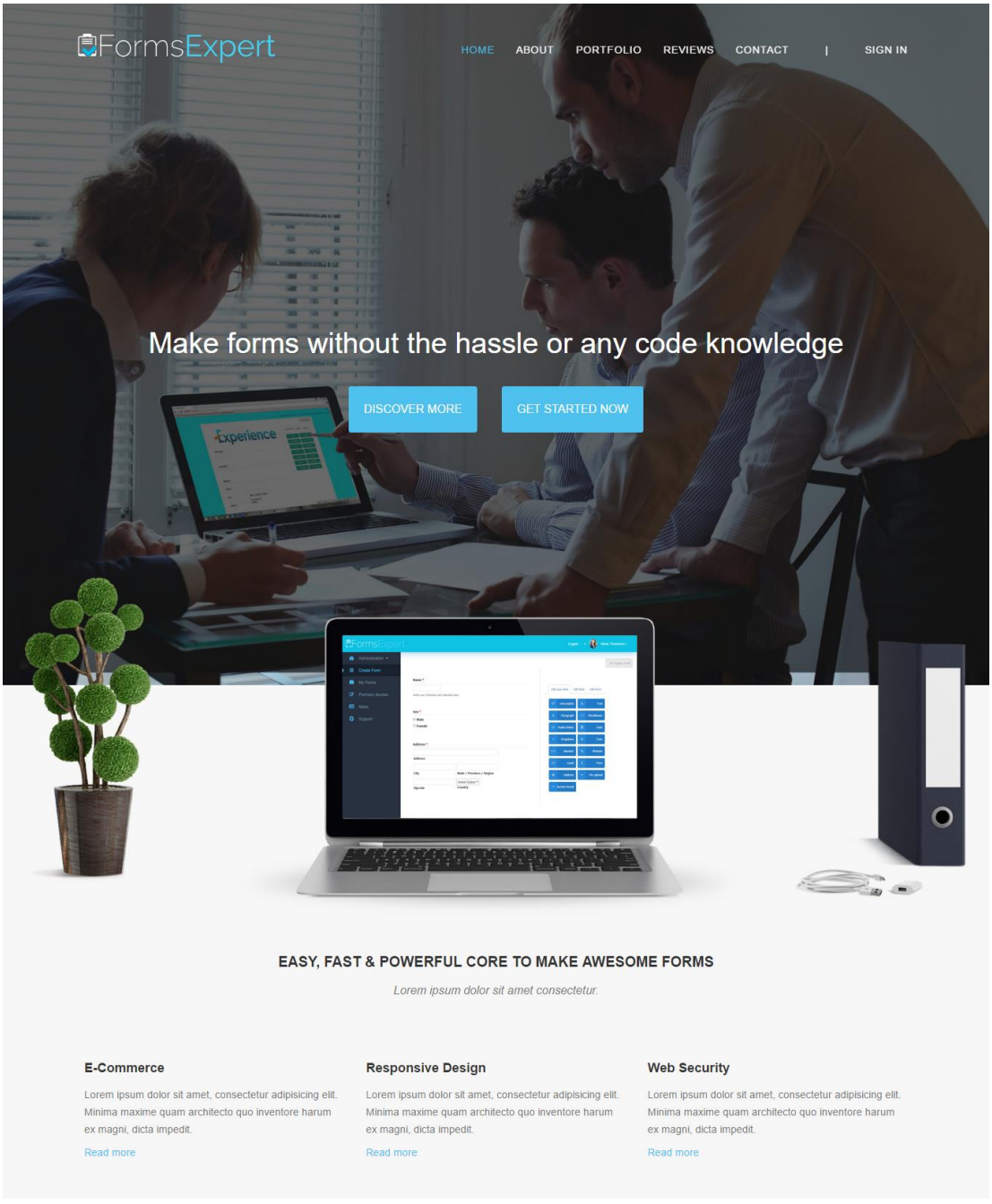
Obr. 13 Případy užití správy administrace

6.7 Grafický návrh

Základní grafický návrh byl navržen grafikem, který poskytl základní layout dvou částí webu – veřejné a neveřejné. Toto provedení bylo dodáno jako šablona vytvořená v programu Adobe Photoshop, na jehož základě byla provedena implementace v HTML za použití Bootstrapu, jelikož s jeho použitím lze vytvářet jednoduše responzivní design, který je v současné době pomalu standardem u většiny webů. Celý web je částečně inspirován současným trendem flat a materiál designu, především je snaha využívat co nejvíce jednolité barvy a minimalistické pojetí webu.

Web je rozdělen na dvě části – veřejnou a neveřejnou. Veřejná část slouží jako uvítací obrazovka pro nepřihlášeného uživatele s informacemi o celém projektu. Jelikož jeho účelem je poskytnout dostatečné informace o projektu a netýká se samotné práce v aplikaci, je tato část webu graficky odlišná od ostatních dvou modulů. V horní části obrazovky je v levém rohu umístěno logo apli-

kace a vedle něj menu, které pouze návštěvníka posunuje dolů po sekcích dané stránky. Zároveň zde nechybí možnost kliknout na přihlašovací tlačítko, které návštěvníka přesměruje na přihlašovací formulář (případně registrační formulář)



Obr. 14 Ukázka informační části webu pro nepřihlášeného uživatele

Jakmile se uživatel přihlásí (zaregistruje), dostává se do neveřejné části webu, která je již graficky přizpůsobena pro práci v aplikaci. Ukázky lze vidět na

obrázcích níže, kde je zachycena práce v aplikaci. Je zde snaha o co nejintuitivnější ovládání, které nebude uživatele aplikace rozptylovat. V horní části webu je panel jednolité světle modré barvy, který obsahuje vlevo logo aplikace a vpravo menu pro změnu jazyka webu a uživatelskou administraci.

V levé části aplikace se nachází menu šedé jednolité barvy, které slouží pro samotnou práci v aplikaci. Uživatel zde může zobrazit novinky, využít podporu moderátorů, zaplatit si prémiový přístup, nastavit si další nastavení formulářů a samozřejmě taky vytvořit nový formulář a zobrazit si již existující formuláře.

V případě, že uživatel přistupuje k aplikaci pomocí mobilního zařízení, je menu díky Bootstrapu skryto, aby se efektivně využila malá plocha těchto zařízení. Přístup k menu je zajištěn tlačítkem, které se nachází v pravém horním rohu. Uživateli na něj může kliknout a menu se mu zobrazí.

6.8 TwiGrid tabulky

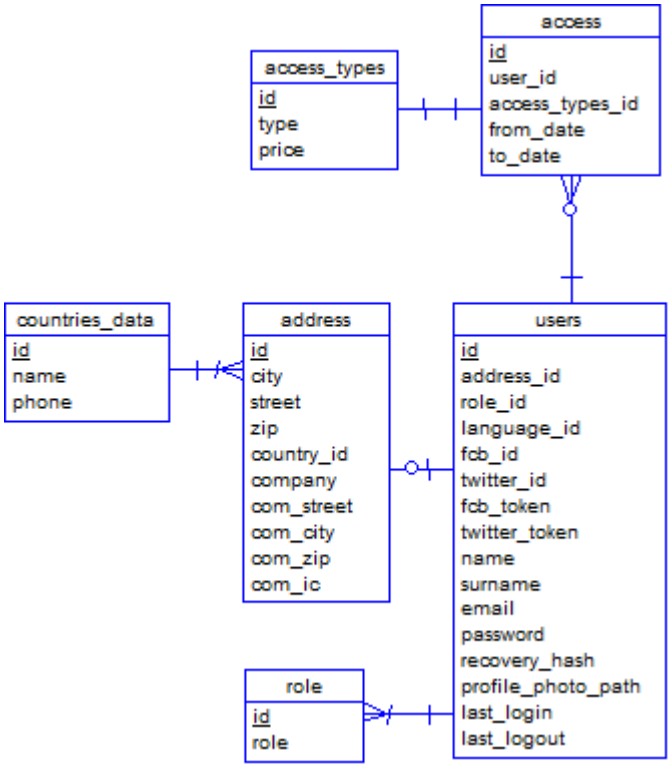
Pro jednotný vzhled výpisu obsahu, který má být uživateli přehledně zobrazen, je v rámci celé aplikace použito rozšíření TwiGrid tabulek, které mají tu výhodu, že jejich vzhled se nadefinuje pouze v jednom CSS souboru a projeví se napříč celou aplikací. Další výhodu je přehlednost, neboť se všechny tabulky ukládají do složky *app/grids* a v šabloně je již stačí vykreslit stejným způsobem jako formuláře. TwiGrid tabulky vycházejí z Twitter Bootstrap 3, takže je zajištěna responzivnost zobrazovaného obsahu, čímž je vývojáři ušetřen čas, který může věnovat vývoji dalších částí webu.

TwiGrid využívá mnoha metod, z nichž jsou pro samotné fungování nejdůležitější metody `build()` a `dataLoad()`. První z nich má na starost především definování sloupců, akcí a možností dané tabulky, na jejichž základě se dále mohou definovat další metody. Druhá metoda zajišťuje samotné získání dat z databáze. Přehledný a detailní popis ostatních metod a všech možností, které lze v tomto rozšíření použít, se nachází na domovské stránce tohoto rozšíření.

6.9 Návrh databáze

Entitně relační diagram, znázorňující databázovou strukturu, je vytvořen pomocí nástroje PowerDesigner. Samotná implementace databáze je v databázovém systému MySQL, pro potřeby Nette je použit formát InnoDB.

Ačkoliv je databáze (až na jednu tabulku) celistvá, je v této kapitole celý návrh databáze pro lepší pochopení a z důvodu přehlednosti rozdělen na několik logických částí, které jsou zobrazeny zvlášť. Toto rozdělení zároveň vystihuje hlavní životní cyklus celé webové aplikace. Celistvý návrh databáze se všemi 27 tabulkami lze nalézt v příloze A, kde jsou tyto části pro lepší orientaci navíc barevně odlišeny.

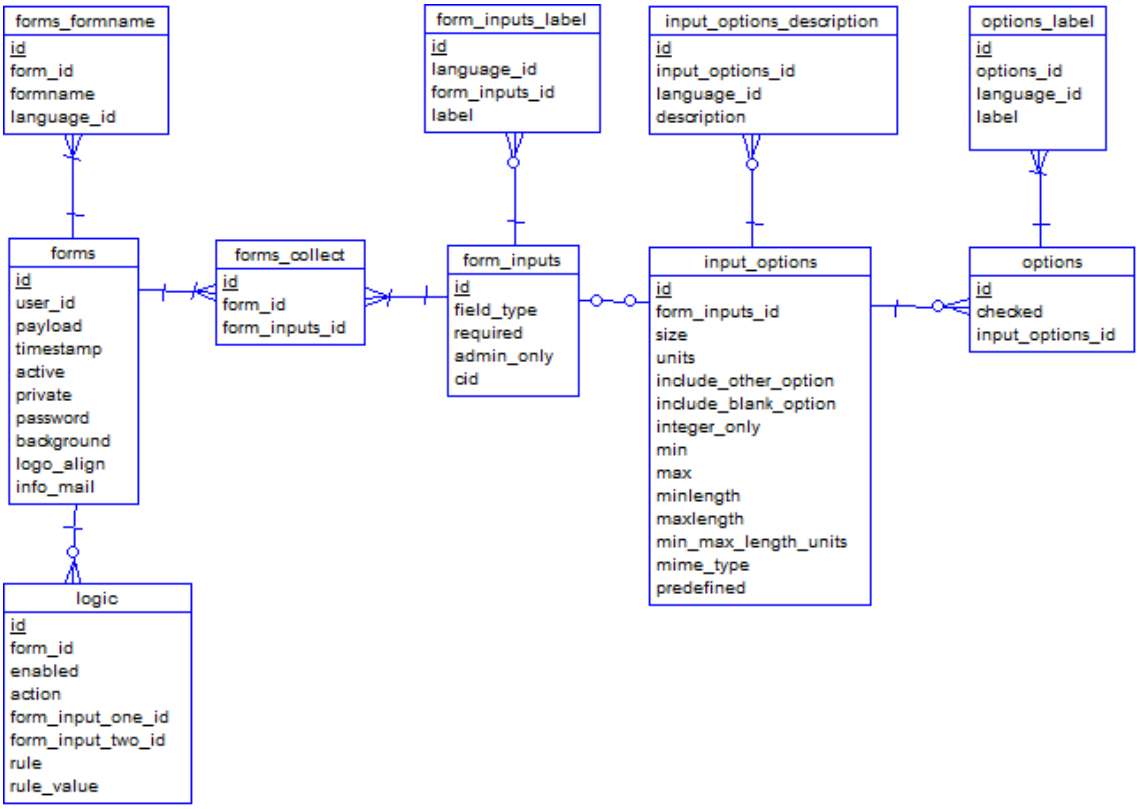


Obr. 15 ERD s tabulkami týkajícími se uživatele

Údaje, týkající se uživatele, se ukládají do tabulky **Users**. Mezi povinné parametry patří jméno (*name*), příjmení (*surname*), email a heslo (*password*). Jelikož aplikace umožňuje i registraci uživatele pomocí sociálních sítí, jsou v tabulce Users uloženy ještě hodnoty o těchto sociálních sítích. *Recovery_hash* obsahuje textovou hodnotu nutnou pro obnovu hesla, dále je zde uložena cesta k obrázku (*profile_photo_path*) a informace o posledním přihlášení a odhlášení (*last_login* a *last_logout*). Uživatelova adresa se ukládá do tabulky **Address** s využitím cizího klíče *address_id*. V této tabulce jsou kromě osobní adresy i data o firemní adrese (začínající na *com_*) v případě, že se jedná o společnost. Tato tabulka je navíc propojená s tabulkou **Countries_data**, pomocí cizího klíče *country_id*, kde se nachází jména a telefonní předvolby daných zemí.

Každý registrovaný uživatel získá při registraci uživatelská práva, jejichž hodnota je pomocí cizího klíče *role_id* uložena v tabulce **Role**.

V případě, že si uživatel předplatí placený přístup ke službám, uloží se do tabulky **Access** hodnoty, které určují typ přístupu, jeho cenu a dobu, po kterou přístup platí. Tato tabulka je propojena s tabulkou **Access_types**, kde jsou uloženy hodnoty o typech přístupů.



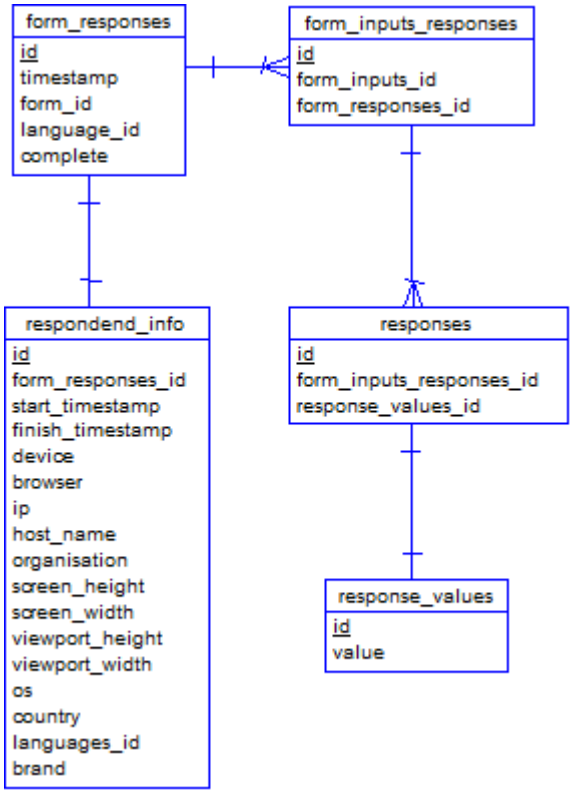
Obr. 16 ERD s tabulkami týkajícími se formulářů

Tabulka **Forms** obsahuje základní informace o formulářích – kdy byl formulář vytvořen (*timestamp*), dále boolean hodnoty, definující, zda je aktivní (*active*), zda je soukromý, tedy jestli jej může zobrazit někdo jiný než vlastník (*private*). V případě, že je soukromý, tak je zde ještě uloženo heslo (*password*), které je potřebné pro přístup těm lidem, kteří toto heslo znají od vlastníka daného formuláře. Zároveň formulář může mít nastaveno pozadí (*background*), pozici loga (*logo_align*). Boolean hodnota *info_mail* určuje, zda při odpovědi na formuláři má jeho majitel dostat email o této nové odpovědi či nikoliv. Pro případ, že by uživatel chtěl svůj existující formulář někdy v budoucnu editovat, nachází se zde hodnota *payload*, která obsahuje JSON (JavaScript Object Notation) objekt s daty o formuláři. Tato data jsou nutná pro případné znovuvytvoření formuláře ve FormBuilderu. Každý formulář může mít několik jazykových verzí, proto se v tabulce **Forms_formname** nachází jeho název (*formname*) a id jazyka (language_id).

Každý formulář je složen z různých prvků, které jsou uloženy v tabulce **Form_inputs**. Jelikož jich může být více, jsou s formulářem propojeny pomocí tabulky **Forms_collect**. Každý prvek v tabulce Form_inputs má svůj typ (*field_type*), boolean hodnotu *required*, která určuje, zda musí být prvek vyplněn, a speciální identifikační kód pozice prvku ve formuláři (*cid*), potřebný pro případ editace ve FormBuilderu. Opět díky možnosti vícejazyčnosti formuláře je potřeba ukládat jazykovou verzi popisků daných prvků, proto v tabulce

Form_inputs_label je popisek (*label*) spolu s id jazyka, ve kterém daný popisek je napsán. Každý prvek má všeobecné vlastnosti, které se týkají jeho velikosti (*size*), jednotek (*units*), intervalu vkládaných hodnot (*min*, *max*, *minlength*, *maxlength*), MIME typ souboru (*mime_type*), případně předdefinovanou hodnotu (*predefined*). Tyto vlastnosti prvku jsou uloženy v tabulce **Inputs_options**. Prvek může obsahovat i popisek, který může být vícejazyčný, proto je zde opět tabulka **Inputs_options_description**, která zajišťuje vícejazyčnost. V některých případech obsahuje prvek kromě všeobecných vlastností ještě několik dalších specifických vlastností. To se týče především prvků typu checkbox, radio button a podobně, kde se daný prvek skládá z více částí (zaškrtávacích možností), které mají další vlastnosti – například popisek (muž, žena), zda daná možnost má být při zobrazení formuláře automaticky označena (*checked*). Opět z důvodu vícejazyčnosti je potřeba tyto popisky ukládat v tabulce **Options_label**.

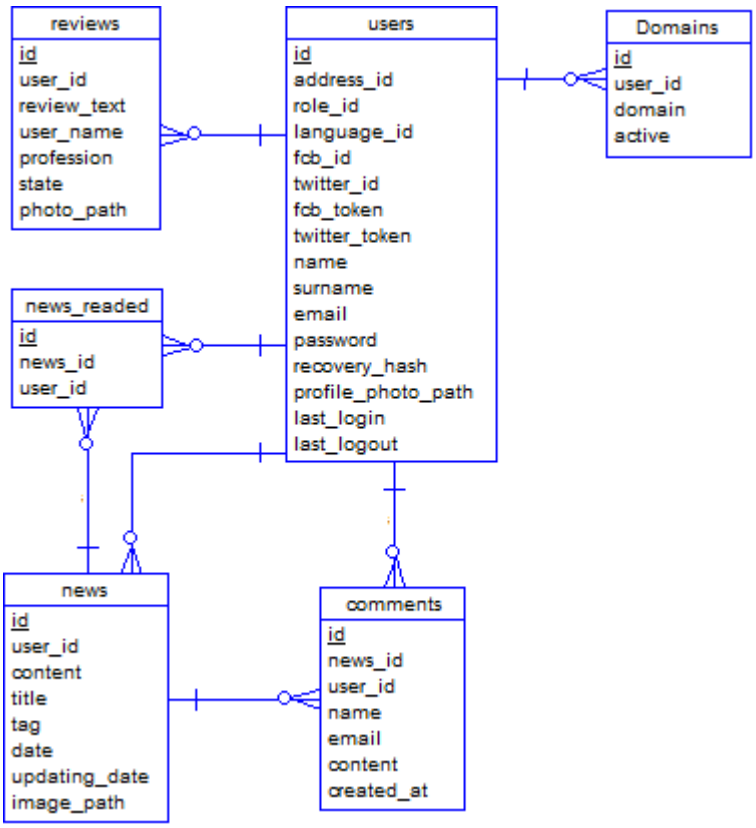
Poslední tabulka **Logic** slouží k tomu, aby se zobrazovaly prvky ve formuláři na základě uživatelem nastavených pravidel. Do hodnoty *action* se ukládá hodnota show nebo hide pro zobrazení nebo skrytí prvku (*input_one_id*) v případě, že je splněna podmínka druhého prvku (*input_two_id*) na základě typu pravidla (*rule_option*) a případně i vztažené hodnoty splňující pravidlo (*rule_value*).



Obr. 17 ERD s tabulkami týkajícími se odpovědí na formulář

Ve chvíli, kdy někdo na vytvořený formulář odpoví, tak se do tabulky **Form_responses** uloží čas odpovědi (*timestamp*), id formuláře (*form_id*), jazyková verze odpovědi a boolean hodnota *complete*, která nabývá hodnoty *false*, dokud není odpovězeno na celý formulář. To se týče především situace, kdy je formulář rozdělen na několik stran a odpovědi se odesílají postupně. Až při zodpovězení poslední stránky formuláře se mění hodnota na *true*, čímž je odpověď kompletní. Zároveň se do tabulky **Respondent_info** uloží informace o respondentovi – kdy začal odpovídat na formulář (*start_timestamp*), kdy odeslal formulář (*finish_timestamp*), informace o zařízení, ze kterého byl formulář vyplňován (*device*), IP adresa, země, šířka obrazovky a prohlížeč.

Odpovědi se ukládají na základě prvků formuláře, a to pomocí tabulky **Form_inputs_responses**. Jelikož odpovědí na daný prvek může být více (například u adresy obsahuje prvek hodnoty město, ulice, číslo popisné, PSČ, země), je potřeba využít ještě tabulky **Responses**, která sdružuje odpovědi na daný prvek. Samotné hodnoty jsou již uloženy v tabulce **Response_values**.



Obr. 18 ERD s tabulkami týkajícími se novinek, domén a komentářů

V tabulce **News** se ukládá id uživatele, který vytvořil nový příspěvek do novinek (*user_id*), její titulek (*title*), obsah (*content*), tag a adresa obrázku, který je použit (*image_path*). Dále se zde ukládá informace o čase, kdy byl příspěvek přidán a případně následně upravován (*date* a *updating_date*). V tabulce **Comments** jsou ukládány informace o komentářích. V případě komentáře od registrovaného uživatele je zde uloženo jeho id, jméno (*name*), email a obsah komentáře (*content*) s časem přidání samotného komentáře (*timestamp*). Tabulka **News_readed** obsahuje id přečtených komentářů daným uživatelem. Tato tabulka slouží především k informování uživatelů, že za určitou dobu od jejich posledního přihlášení přibylo několik novinek, které ještě nepřečetli.

V tabulce **Reviews** jsou uloženy recenze od uživatelů, do níž se ukládá text recenze (*review_text*), jméno člověka, který poskytl recenzi (*person_name*), jeho profese (*profession*), stát a odkaz na jeho fotografii (*photo_path*).

Dále v tabulce **Domains**, se ukládají informace o doménách. Id uživatele (*user_id*), který ji přidal, adresa samotné domény (*domain*) a boolean hodnota *active*, která určuje, zda je přístup z domény aktivní nebo neaktivní.

Další, velmi důležitou tabulkou, která je vyobrazena v ERD diagramu v příloze A, je tabulka **Languages**, která obsahuje dostupné jazyky, především pro případ vícejazyčných formulářů a volby jazyka pro jazykovou verzi celé webové aplikace.

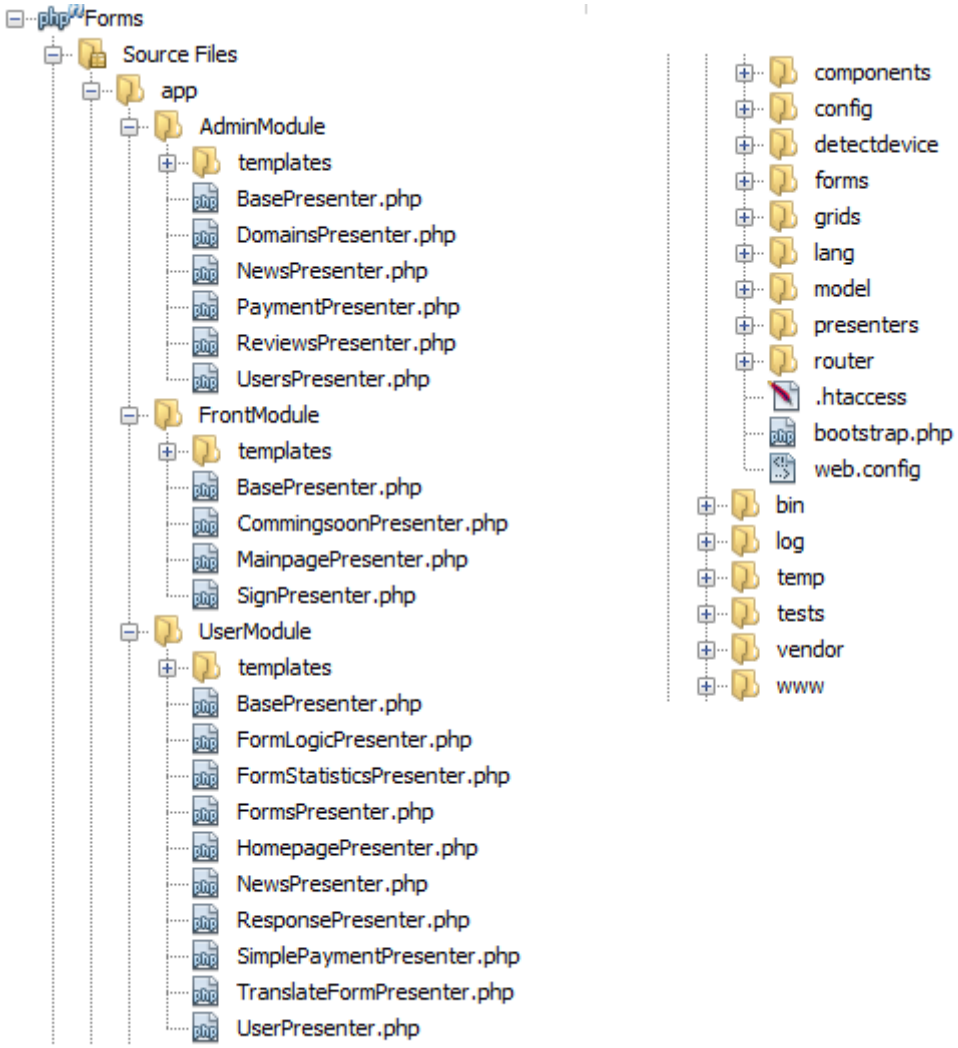
7 Implementace s využitím Nette frameworku

Pro vývoj aplikace byl využito vývojové prostředí NetBeans IDE 8.0.2, které má již implicitně od verze 7.4 zabudovaný PHP plugin přímo pro Nette framework.

7.1 Adresářová struktura projektu

Jelikož vyvíjená aplikace se postupně během vývoje rozvíjí a do budoucna se bude i nadále rozvíjet, bylo potřeba adresářovou strukturu již na začátku upravit, respektive rozdělit presentery a šablony do tří logických modulů, aby byl celý projekt přehlednější a lépe se spravoval.

Kromě adresářů modulů, kde jsou uloženy presentery a šablony, stojí za povšimnutí adresář *forms*, který v sobě obsahuje znovupoužitelné formuláře, které lze, jak již název vypovídá, použít na více místech v aplikaci, aniž by bylo potřeba je opětovně programovat v presenterech. Dále je zde adresář *model*, který obsahuje modely, které mají na starost komunikaci s databází. Adresář obsahuje více modelů, které jsou opět logicky oddělené, aby bylo docíleno větší přehlednosti. Celá adresářová struktura tedy vypadá následovně.



Obr. 19 Adresářová struktura projektu

7.2 Uživatelské role

V současné době se v aplikaci využívá pouze dvou rolí – uživatel a administrátor. Aplikace, jak již bylo zmíněno výše, je rozčleněna na tři moduly. Do *FrontModule* má přístup nepřihlášený uživatel, do ostatních dvou modulů je řešen na základě uživatelských rolí, které jsou uloženy v sessions přihlášeného uživatele. To v tomto případě znamená, že uživatelé s rolí uživatel a administrátor mají plný přístup do modulu *UserModule*, ale do modulu *AdminModule* má přístup pouze uživatel s administrátorskými právy.

Kontrola přístupů do daných modulů je řešena již v *BasePresenterech* konkrétních modulů v metodě `startup()`, díky čemuž se ještě před samotným zpracováním jakéhokoliv presenteru v daném modulu ověřuje, zda má daný uživatel povolen přístup. Pokud přístup nemá, je okamžitě přesměrován na *User-*

Module, který jej případně (jestliže uživatel není přihlášen) dále přesměruje na *FrontModule*.

V budoucnu, jakmile se začne aplikace dále rozrůstat, bude nutné přidat další uživatelské role. Na jejich základě by přibyly pouze nové moduly, nastavila by se oprávnění do těchto modulů a naimplementovaly se nové presentery a šablony.

7.3 Registrace uživatele

Vezmeme-li popis fungování a samotné implementace aplikace z pohledu životního cyklu práce v aplikaci, první, co uživatel musí udělat, aby v dané aplikaci mohl pracovat, je jeho registrace a přihlášení do aplikace, kterou má na starost *SignPresenter* ve *FrontModulu*.

Uživatel má na výběr ze dvou možností, jak registraci provést. Může se registrovat pomocí registračního formuláře, nebo využije registraci pomocí veřejných údajů z Facebooku a Twitteru.

V případě první možnosti, tedy plnohodnotné registrace přes registrační formulář, je formulář zobrazen pouze jednosloupcově, kde uživatel zadá svoje údaje, jako je jméno, příjmení, email, heslo. V případě, že se jedná o společnost, která chce danou aplikaci využívat, je potřeba v horní části tohoto formuláře zvolit v select boxu, že se jedná o společnost. V tom okamžiku se v registračním formuláři zobrazí ještě pravý sloupec, kde je ještě nutno vyplnit adresu sídla dané společnosti.

Aby toto dynamické zobrazování a skrývání formuláře fungovalo, je potřeba použít dvou metod – `addCondition()` a `toggle()`. Funguje to tak, že metoda `addCondition()` umí přijímat stejné argumenty, jako základní validační metoda `addRule()`. Pokud je tedy podmínka v `addCondition()` splněna, tak metoda `toggle()` s argumentem id atributu ve výsledném HTML kódu tento element s tímto id dynamicky zobrazí nebo skryje právě v závislosti na podmínce, která je definována `v addCondition()`. V praxi to lze naimplementovat následovně:

```
$this->addSelect('userType', '', array('0' => 'Free', '1' => 'Paid'))
                    ->setDefaultValue($defaultValue)
                    ->addCondition($this::EQUAL, 1)
                    ->toggle('fin');
```

Aby byl druhý sloupec formuláře obalen elementem `<div id=fin>…</div>`, je potřeba seskupit prvky formuláře do skupiny za použití metody `addGroup()` a nastavit vlastnosti, respektive id elementu div, pomocí metody `setOption()`. To lze učinit následovně:

```
$fin = $this->addGroup('Company address')->setOption('container', 'div
id=fin');
```

Jelikož při registraci uživatel zadává svůj email, který bude spolu s heslem sloužit pro budoucí přihlášení, je nutné, aby byl email unikátní. Proto je potřeba před samotným zaregistrováním uživatele zkontrolovat, zda se již email nenachází v databázi. V případě, že se email v databázi již nachází, je uživateli zobrazena hláška, že musí použít jiný email. Samotná validace, zda je email ve správném tvaru, je ponechána na Nette frameworku. Při tvorbě registračního formuláře stačí pouze přidat prvku Email pravidlo `Form::EMAIL`

Druhou možností, jak se může uživatel registrovat, je buď pomocí Facebooku, nebo Twitteru za použití OAuth autorizace. V tomto případě nemusí uživatel ručně zadávat svoje osobní údaje, ale vystačí si pouze se dvěma kliknutími (pokud je tedy v danou chvíli přihlášen k dané sociální síti, v opačném případě se do ní musí ještě přihlásit). Pro úspěšnou registraci stačí pouze jméno, příjmení a email, které jsou po úspěšném ověření uživatele a propojení s aplikací poskytnuty. Uživatel je přesměrován na web dané sociální sítě, kde se mu zobrazí potvrzovací dialog, že chce aplikaci poskytnout svoje jméno, příjmení a email (případně lze poskytnout i další veřejně přístupné informace jako profilový obrázek a podobně), a je opět přesměrován zpět na naši aplikaci, které jsou tyto údaje předány. V případě, že se již v minulosti takto zaregistroval a potvrdil přístup aplikace ke svým údajům, tak se mu již daný potvrzovací dialog nezobrazí, ale namísto toho je přesměrován na danou sociální síť a následně přesměrován zpět na naši webovou aplikaci, do které je okamžitě přihlášen.

Z pohledu aplikace bylo nutné pomocí Composeru stáhnout rozšíření *kdyby/facebook* a *ipub/twitter* a na daných sociálních sítích bylo potřeba zaregistrovat naši webovou stránku jako aplikaci (Facebook App / Twitter Apps) v rámci developerské části dané sociální sítě, kde bylo potřeba vyplnit doménu naší aplikace, čímž říkáme, z jaké URL lze přihlašování použít. To znamená, že v případě přihlašování z jiné (neregistrované) domény by přihlášení neproběhlo. Následně jsme získali identifikační číslo a speciální tajný kód, který bude nutný použít v rámci konfigurace dané přihlašovací komponenty. Tyto dva údaje je potřeba nakonfigurovat v souboru *config.neon*, neboť bez nich by celá autorizace nemohla proběhnout. Zároveň je potřeba zaregistrovat tyto rozšíření pomocí *extensions*.

Jako poslední krok pro úspěšnou registraci a přihlašování pomocí těchto dvou sociálních sítí už zbývalo pouze v *SignPresenteru* vytvořit metodu `createComponentFbLogin()`, která volá metody z modelu *UsersModel*, kde bylo nutné naimplementovat samotné uložení dat registrovaného uživatele do databáze a jeho následné přihlášení. K tomu slouží metody:

- `findByFacebookId($user)` – vyhledá uživatele pomocí jeho facebookového id a vrátí výsledek. Slouží pro ověření, zda je uživatel již registrován. Pokud ne, je volána následující metoda
- `registerFromFacebook($user, $me)` – na základě hodnot v proměnné `$me`, kterou nám vrací Facebook po úspěšném ověření uživatele, je uživatel uložen do databáze.

- `updateFacebookAccessToken($user, $token, $email)` – Pokud dojde ke změně tokenu již dříve zaregistrovaného uživatele, je potřeba tento token aktualizovat v databázi.

V případě úspěšné registrace a pozdějšího přihlašování pomocí sociálních sítí, dojde po přesměrování na danou sociální síť k pokusu o ověření daného uživatele a v případě úspěšného ověření je následně uživatel opět přesměrován zpět na naši webovou aplikaci spolu s jeho id a unikátním tokenem, který byl vygenerován sociální sítí při první registraci uživatele. To stačí pro uživatelovu autorizaci, neboť tyto hodnoty jsou již od jeho registrace uloženy v databázi. Token v tomto případě nahrazuje heslo uživatele a jeho login reprezentuje jeho id na sociální síti.

Jelikož v tomto případě uživatel při registraci neposkytuje příliš mnoho údajů, má možnost si kdykoliv svoje kontaktní údaje ve své administrativní části vyplnit a taktéž si může nastavit heslo pro případ, že by se chtěl moci přihlásit i za pomocí zadání emailu a hesla.

V případě, že uživatel zapomene heslo nebo jej nemá, neboť se registroval pomocí sociálních sítí a nyní se pomocí nich z nějakého důvodu nemohl přihlásit, může si jej nechat resetovat. Při požadavku na nové heslo musí uvést svůj email, použitý při registraci. Pokud je email uložen v databázi, obdrží na něj námi vygenerovanou zprávu s odkazem, který obsahuje automaticky vygenerovaný token, nutný pro resetování hesla, spolu s daným emailem, na který byl token zaslán. Tento token je vytvořen pomocí metody `time()` spolu s hodnotou náhodného čísla `rand()`. Náhodné číslo je zde použito z důvodu, aby se nedalo jednoduše odhalit vygenerovaný tokenu. Tato hodnota je následně zašifrována pomocí algoritmu SHA1 a uložena do tabulky *users*. V případě, že token s emailem nesouhlasí s hodnotami v databázi, není možné změnu provést a uživatel je okamžitě přesměrován na *recover* šablonu. V opačném případě, kdy token spolu s emailem souhlasí, je uživateli zobrazen formulář, kde stačí vyplnit dvakrát nové heslo a uložit změnu. Pokud se hesla navzájem shodují, je toto heslo zašifrováno a uloženo do databáze a uživatel je automaticky přihlášen.

7.4 FormBuilder

Aby FormBuilder mohl být vykreslen, je potřeba vložit na požadovanou stránku jednoduchou funkci, která jej zavolá a vykreslí v požadovaném div elementu, který se na stránce nachází. V tomto případě v div element s id `fb-main`.

Pokud bychom chtěli, aby byly ve FormBuilderu zobrazeny již nějaké předdefinované prvky, stačí v atributu `bootstrapData[]` vložit informace o prvcích v JSON formátu. V ukázce níže je defaultně zobrazen prvek formuláře typu logo, kde si uživatel může nahrát do hlavičky formuláře obrázek, který chce použít jako své vlastní logo.

Kdykoliv proběhne jakákoliv změna, ať už se jedná o přidání nebo odebrání prvku formuláře, je automaticky volána funkce `fb.on('save', functi-`

`on(payload)`, která ukládá do skrytého textového pole `payload` výsledný JSON se všemi informacemi, které se týkají všech uživatelem vytvořených prvků formuláře. Tento JSON objekt je důležitý pro následné zpracování a uložení samotného formuláře do databáze.

```
$(function(){
      fb = new Formbuilder({
        selector: '#fb-main',
        bootstrapData: [
          {
            "label": "Load your logo/image in the Edit form tab.",
            "field_type": "logo",
            "required": false,
            "field_options": {},
            "cid": "c1"
          }
        ]
      });

      fb.on('save', function(payload){
        $('input[name=payload]').val(payload);
      })
    });
```

FormBuilder je velmi modulární, takže pro vytvoření nového prvku je potřeba pouze zaregistrovat nový prvek a nastavit mu vlastnosti.

```
(function () {
    Formbuilder.registerField('text', {
        order: 10,
        view: "<input type='text' class=
               'rf-size- <%=
               rf.get(Formbuilder.options.mappings.SIZE) %>'/>",
        edit: "<%= Formbuilder.templates['edit/size']() %>\n
          <%= Formbuilder.templates['edit/min_max_length']() %>",
        addButton: "<span class='fa fa-font'>Text"</span>,
        defaultAttributes: function (attrs) {
            attrs.field_options.size = 'small';
            return attrs;
        }
    });
}).call(this);
```

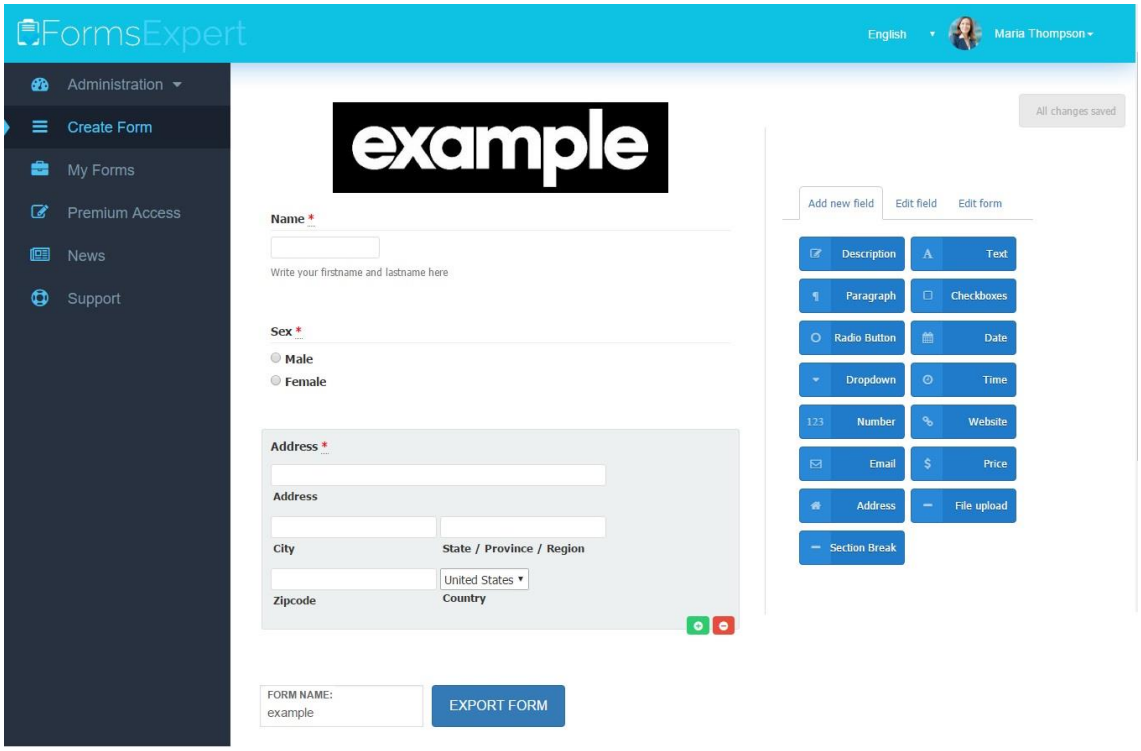
Funkce výše, vzato postupně, má za úkol přiřadit nově registrovanému prvku pořadí, ve kterém má být zobrazen v pravém menu (index `order`). Dále je nutné do indexu `view` uložit HTML zápis daného prvku, aby mohl být vykreslen v levé části FormBuilderu. Zde se tedy jedná o input typu textové pole, který má nastavenou proměnlivou třídu `rf-size-`. Tato třída se mění na základě získání hodnoty ze šablony vlastností.

V případě, že má prvek možnost, aby mu mohly být upravovány vlastnosti, jako je například velikost, maximální počet znaků a podobně, je potřeba nastavit v indexu `edit` některou z již existujících šablon vlastností (templates). `AddButton` slouží k vykreslení tlačítka v pravém menu. Opět je zde využito HTML zápisu, díky němuž nastavujeme ikonu fontAwesome a samotný název prvku, aby uživatel intuitivně na první pohled věděl, jaký prvek toto tlačítko reprezentuje.

Jako poslední je potřeba definovat defaultní vlastnosti prvku, ty jsou uloženy v indexu `defaultAtributes`. V této ukázce je mu nastavena malá velikost (small), kterou samozřejmě může uživatel v nastavení prvku následně změnit.

Zmíněné šablony vlastností, použité pro vlastnosti prvků a ovlivňování jejich vzhledu, jsou implementačně velmi podobné registraci prvků. V případě vlastností, které se týkají především výsledného vzhledu prvku, což je například právě jeho velikost, je tomuto prvku přidána CSS třída. Proto je nutné vždy těmto třídám následně nastavit tyto vlastnosti v CSS stylech.

Jelikož je FormBuilder navržen tak, aby byl jednoduše přizpůsobitelný, byly pro naše potřeby přidány ještě další prvky spolu se třetí záložkou Edit form, která zahrnuje další možnosti editace formuláře – přidání pozadí (funkce `changeBackground()`) a vložení a zarovnání loga. Taktéž bylo upraveno samotné rozhraní FormBuilderu, kdy se editační menu přesunulo na pravou stranu a změnil se grafický vzhled tlačítek na modrou barvu. Současná podoba je na obrázku níže.



Obr. 20 Ukázka tvorby formuláře v upravené verzi FormBuilderu

7.5 Uložení vytvořeného formuláře

Pokud je uživatel se svým navrženým formulářem spokojen, může si jej uložit. V dolní části pod FormBuilderem se nachází textové pole, do kterého napíše název formuláře a následně se metoda `saveDecodedValues($values)` v *HomepagePresenteru*, která získá JSON objekt od FormBuilderu se všemi informacemi o prvcích formuláře, postará o jeho uložení do databáze.

7.6 Vykreslení vytvořeného formuláře

Další nastavení formuláře se týkají presenteru *FormPresenter*. Ve chvíli, kdy si uživatel formulář uloží, je přesměrován na šablonu *showForm*, která vykreslí ukázku finálního formuláře pomocí metody `createComponentMyForm()`, kde jej vidí tak, jak jej uvidí respondenti. Tato metoda na začátku získá (na základě parametru v URL) všechny jeho prvky z tabulky *forms_collect*, které jsou následně pomocí cyklu foreach vytvářeny. V tomto cyklu se pomocí podmínek zjišťuje, o jaký prvek se jedná. Na základě toho se dále nastavují vlastnosti daného prvku (pokud je vůbec má), čímž je docíleno toho, aby byl formulář vykreslen přesně tak, jak jej uživatel vytvořil.

Jelikož je tato metoda poměrně obsáhlá a velmi složitá, níže je ukázka tvorby pouze jednoduchého prvku Paragraph (textarea), který neobsahuje příliš mnoho vlastností:

```
if ($form_inputs['field_type'] == 'paragraph') { //je paragraph?
   //zjisti, zda je prvek povinny
   $requiredClass = $this->getRequiredClass($form_inputs['required']);
   //vytvori kolem celého prvku element section
   $form->addGroup($labelField)
        ->setOption('description', $description)
        ->setOption('container', Nette\Utils\Html::el('section')
        ->class('group ' . $requiredClass));
    //pridej prvek textarea s jeho id hodnotou
    $form->addTextArea($form_inputs['id']);
    //nastav mu velikost na zaklade jeho vlastnosti
    $input = $form[$form_inputs['id']]->getControlPrototype();
    $input->class($field_options['size'] . '-textarea');
    //ma vlastnost minimálního poctu znaku?
    if ($field_options['minlength'] !== NULL) {
       $form[$form_inputs['id']]
        ->addRule($form::MIN_LENGTH, 'Text must contain at least %d
                  letters.', $field_options['minlength']);
    }
    //ma vlastnost maximálního poctu znaku?
    if ($field_options['maxlength'] !== NULL) {
        $form[$form_inputs['id']]->addRule($form::MAX_LENGTH,
        'Text must contain less than %d letters.',
        $field_options['maxlength']);
    }
}
```

Jelikož formulář může obsahovat více prvků, je nutné, aby jejich jména byla unikátní. Proto je každý prvek pojmenován na základě jeho id v tabulce *form_inputs*. V případě prvku, jako je například adresa, kde je nutno vykreslit tento prvek jako soubor více textových polí, je nutné ještě daným textovým polím přidat k tomuto názvu další text oddělený podtržítkem, který identifikuje, o které textové pole se jedná. Tím je zajištěn unikátní název a zároveň znalost id prvku a typu textového pole. V prvku adresa je například textovému poli, kam má respondent napsat město, přidán identifikační text následovně:

```
    $form->addText($form_inputs['id'] . '_ci', 'City');
```

Důvod, proč je formulář ukládán do tabulek databáze, namísto aby byl uložen pouze JSON výstup, a následně je vykreslován pomocí PHP a ne pomocí JavaScriptu, je z hlediska vývoje prostý. Sice tento postup prodloužil dobu vývoje, avšak díky uložení informací o formuláři do databáze je následně zjednodušena další práce s vytvořeným formulářem. Nespornou výhodou tohoto řešení je

validace vstupů ze strany klienta i serveru. Navíc není potřeba se starat o ošetření vstupních hodnot, neboť o to se stará automaticky Nette framework. Další výhoda tohoto řešení je následné zobrazení a vyhodnocování odpovědí, neboť stačí jednoduchý dotaz na databázi, abychom zjistili, ke kterému prvku formuláře (důležitý je především jeho název, v některých případech i hodnoty prvku) patří konkrétní odpovědi. Další výhodou je i to, že lze jednoduše vytvářet vícejazyčné verze formuláře.

Jinými slovy se FormBuilder stará pouze o to, aby si v něm uživatel vytvořil jednoduše svůj formulář a poskytl nám informace o jeho prvcích a jejich vlastnostech. Zbytek má již na starost naše aplikace.

7.7 Další nastavení vytvořeného formuláře

Nad výsledným formulářem se nachází další možnosti, týkající se rozšířených vlastností, které lze s vytvořeným formulářem dělat. Tyto možnosti vidí pouze přihlášený uživatel, který je vlastník konkrétního formuláře. Kontrola je zajišťována metodou `isOwner($form_id, $responseId)`, která vrací boolean hodnotu na základě porovnání vlastníka daného formuláře a id přihlášeného uživatele. Tuto metodu lze použít i pro zobrazení odpovědí na základě parametru `$responseId`.

Uživatel může například vytvořit novou jazykovou verzi již existujícího formuláře, nastavit formuláři heslo, aby na něj mohli odpovědět pouze respondenti, kteří toto heslo znají. Zároveň lze tento formulář sdílet pomocí emailu, případně nastavit, aby v případě nových odpovědí od respondentů, byl vlastník daného formuláře upozorněn emailem.

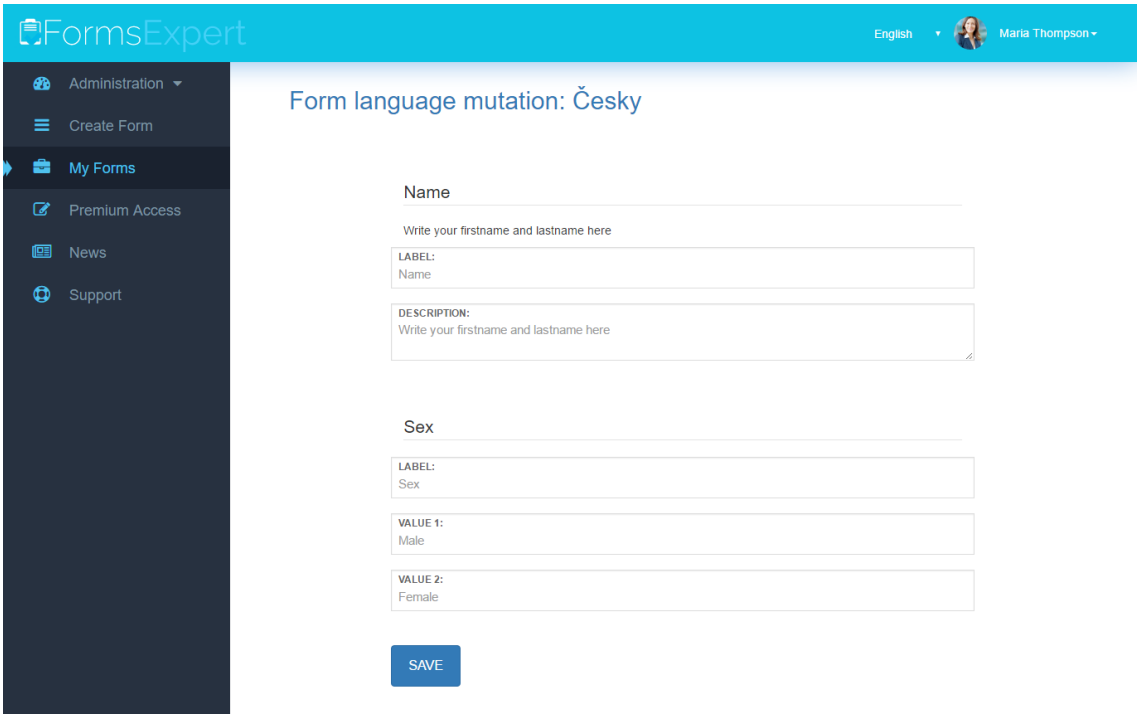
Taktéž lze vytvořit kopii, kterou může uživatel upravit ve FormBuilderu a uložit jej jako novou verzi. Důvod, proč se neupravuje přímo vytvořený formulář a následně se neuloží pouze provedené změny, je ten, že v případě již uložených odpovědí na formulář by mohlo dojít k ovlivnění již existujících odpovědí. Buď by již nebyly vázány na konkrétní prvek (v případě jeho smazání), což by mohlo vést ke ztrátě odpovědí, nebo by se mohlo stát, že v případě změny znění otázky konkrétního prvku by byly odpovědi zkreslené. Ukázkovým příkladem může být změna názvů hodnot dvou checkboxů „žena“ a „muž“ tak, aby byly seřazeny abecedně, tedy změna na „muž“ a „žena“. Nemluvě o situaci, kdy by v případě přidání nového prvku na tento prvek bylo méně odpovědí, což by si uživatel nemusel uvědomit a vedlo by to opět ke zkreslení výstupů.

7.8 Nová jazyková verze formuláře

V případě, že uživatel chce vytvořit novou jazykovou verzi, vybere si v roletkovém menu jazyk, do kterého chce formulář přeložit. Následně je přesměrován na šablonu *translateForm*, kde se mu zobrazí všechny prvky, nutné pro překlad. Vytvoření těchto prvků má na starost *TranslateFormPresenter* a jeho funkce `createComponentTranslateForm()`. Samotný překlad se týká

v podstatě všech textových popisků, které uživatel ručně vypisoval při tvorbě formuláře – otázky, popisky, hodnoty checkboxů a radio buttonů. Každý prvek je reprezentován popisky, které je potřeba přeložit, a následně se pod nimi nachází textová pole, do nichž je nutno vložit jejich překlad. Aby uživatel intuitivně věděl, které textové pole se vztahuje ke kterému popisku, mají tato textová pole uvnitř výchozí text daného popisku (placeholder), který zmizí, jakmile do tohoto pole začne psát překlad.

V případě, že vlastník formuláře vytvoří více jazykových verzí, zobrazí se nad formulářem select box s existujícími jazykovými verzemi, díky čemuž si může respondent mezi nimi přepínat a odpovídat na takovou jazykovou verzi formuláře, které rozumí. Defaultně by se respondentovi měla zobrazit taková jazyková verze (pokud tedy existuje), která odpovídá jeho jazyku uvedeném při registracci, případně jazyku prohlížeče. V opačném případě je verze jazykové mutace nastavena na tu, která byla vytvořena jako první.



Obr. 21 Ukázka překladu formuláře do češtiny

7.9 Sdílení formuláře

Vytvořený formulář může uživatel rozesílat emailem přímo z webové aplikace. Nad vytvořeným formulářem vybere možnost sdílení formuláře a je přesměrován na stránku pro jednoduché rozesílání emailových zpráv. O vykreslení formuláře se stará metoda `createComponentEmailForm()`.

Okno pro rozeslání emailu obsahuje jednoduchý WYSIWYG editor od společnosti TinyMCE, který je volně k použití. V tomto editoru je již defaultně vlo-

žena zpráva, která má být odeslána, spolu s odkazem na uživatelův formulář. Tuto zprávu může uživatel samozřejmě editovat podobným způsobem, jaký zná z různých kancelářských balíků, jako je například MS Word, Libre Office a podobně. V případě zručnějšího uživatele lze tento editor přepnout do HTML kódu a upravit si zprávu ručně.

Implementace tohoto editoru je velmi jednoduchá, do zdrojového kódu je potřeba vložit pouze skript s odkazem na JavaScriptovou knihovnu editoru a následně skript, ve kterém je definováno, který prvek má být vykreslen jako editor.

```
<script src="//cdn.tinymce.com/4/tinymce.min.js"></script>
<script>tinymce.init({ selector:'#message' });</script>
```

7.10 Odpovědi na formulář

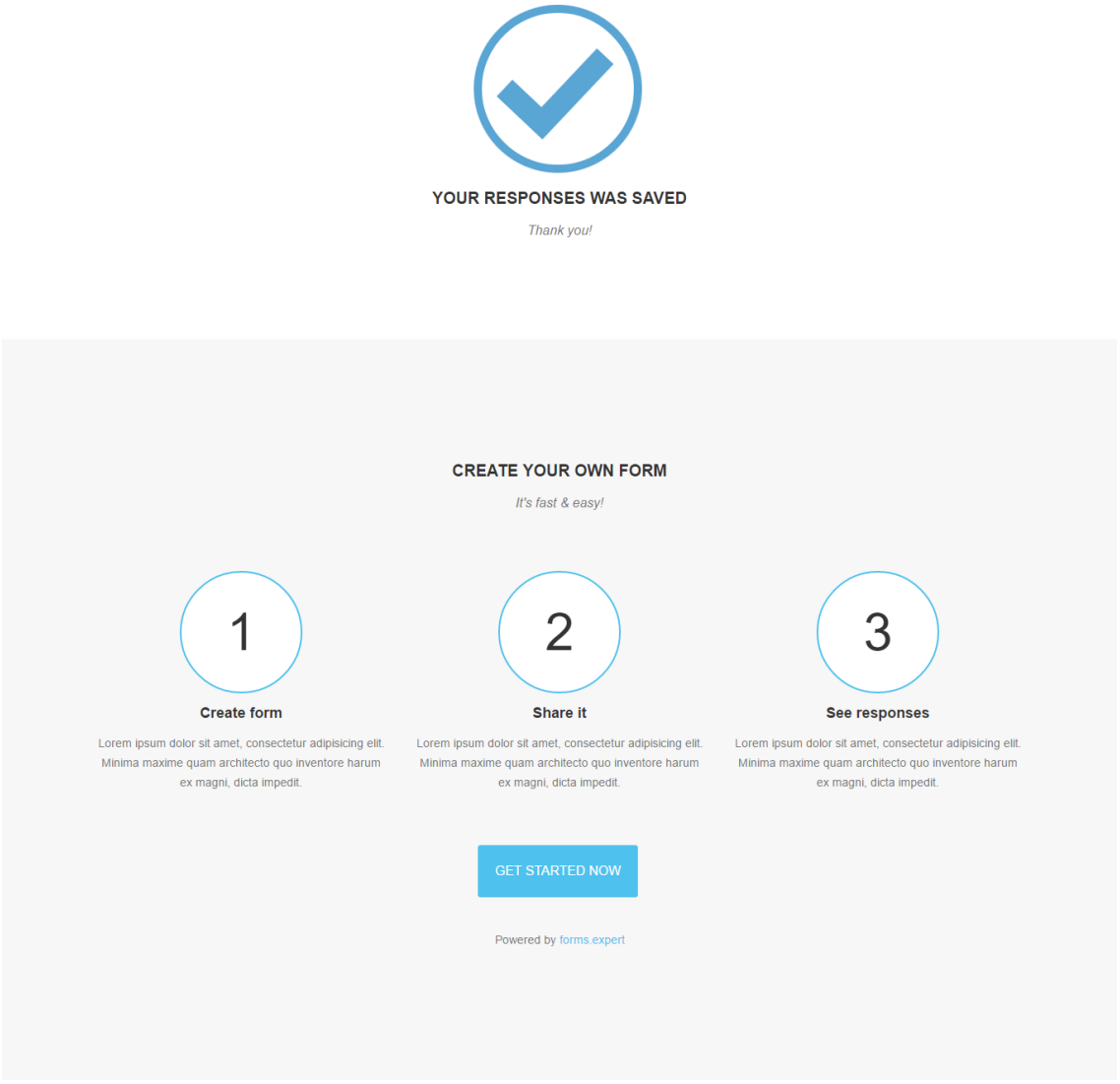
Ve chvíli, kdy začne respondent odpovídat na vytvořený formulář, jsou do databáze pomocí funkce `saveRespondentInfo()` do tabulky *respondent_info* uloženy základní informace o něm (viz kapitola 6.9 Návrh databáze) a především čas, kdy tento formulář zobrazil. V aplikaci je uchováváno id tohoto záznamu, aby bylo možné budoucí updatování hodnoty *finish_timestamp*. To je důležité k budoucím statistikám, kde je vhodné zachycovat dobu vyplňování a taktéž to, zda bylo na formulář kompletně odpovězeno. V opačném případě se hodnota *finish_timestamp* neupdatuje a obsahuje defaultní hodnotu *NULL*. Jakmile respondent odpoví na všechny otázky a odešle formulář, je do databáze uložen čas jeho odeslání na základě id záznamu, který vznikl ve chvíli zobrazení tohoto formuláře.

V případě, že se jedná o formulář, který obsahuje více stran, je hodnota *finish_timestamp* uložena až ve chvíli, kdy je zodpovězeno na poslední část formuláře. Samotné ukládání odpovědí má na starost funkce `formSendSucceeded($form, $values)`.

Abychom bylo možné uložit odpovědi ke správným prvkům formuláře, je nutné kontrolovat jejich název. V některých případech je jejich názvem pouze jejich id, avšak jak již bylo zmíněno výše, v případě některých prvků, které se skládají z více textových polí, je nutno následovně pomocí funkce `explode()` získat id a typ textového pole:

```
$substr = substr($key, -2); //poslední 2 znaky identifikujici prvek
$id = explode('_', $key); //oddel id od identifikačních znaku prvku
    if ($substr == 'dd') { //je substr textovym polem dne?
        $day = $value; //uloz hodnotu do promenne $day
        $day_id = $id[0]; //uloz id prvku do promenne $day_id
    }
```

Pokud vše proběhne v pořádku, je uživatel přesměrován na šablonu *success*, kde se mu zobrazí poděkování za odpověď, základní informace o této webové aplikaci a je mu taktéž nabídnuta možnost si vytvořit svůj vlastní formulář.

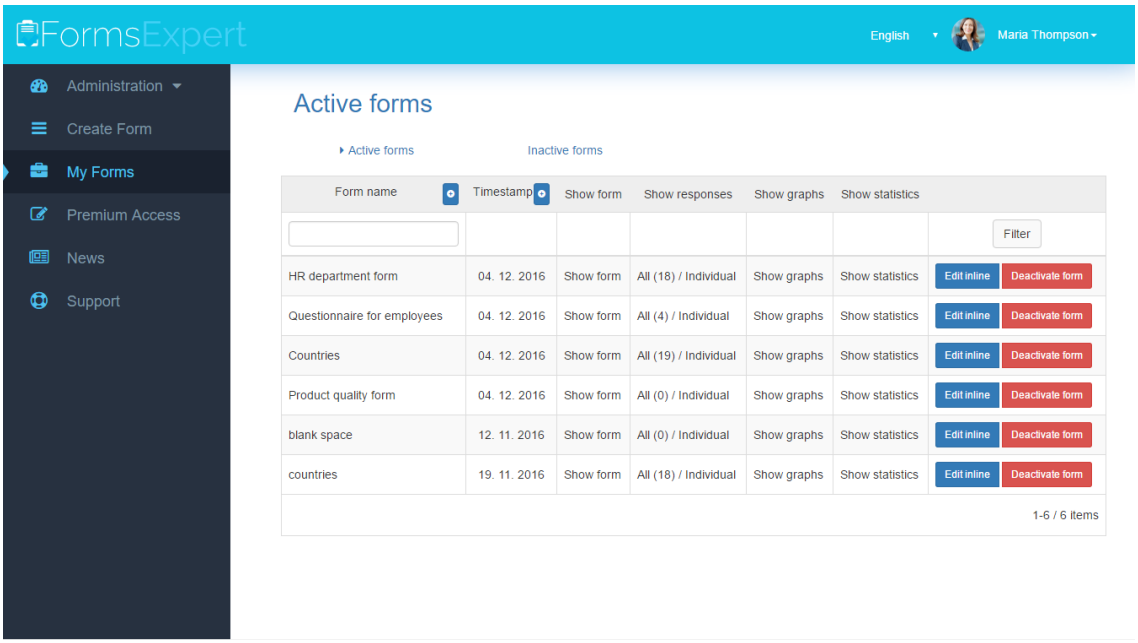


Obr. 22 Ukázka stránky, která se zobrazí po úspěšném odeslání formuláře

7.11 Zobrazení vytvořených formulářů

Uživatel si samozřejmě může vytvářet více než jeden formulář. Pro zobrazení základního přehledu o jeho vytvořených formulářích využije v menu sekci My Forms, kde má TwiGrid tabulku se základním přehledem o jeho formulářích. V této sekci může přepínat mezi aktivními a neaktivními formuláři (Active forms / Inactive forms). V tabulce vidí seznam vytvořených formulářů

a u každého z formulářů má možnost zobrazit si odpovědi – jak všech odpovědí na formulář, tak i individuální na konkrétní prvek formuláře. Zároveň je v této tabulce možnost zobrazit grafické vyhodnocení odpovědí na celý formulář a základní metrické statistiky o odpovědích. Kromě toho lze v této tabulce využít akce na přejmenování a deaktivaci formuláře. Deaktivace formuláře je vhodná ve chvíli, kdy uživatel již nechce, aby na daný formulář mohli respondenti odpovídat. Obsah tabulky lze zároveň filtrovat podle názvu formuláře, případně seřadit dle abecedy nebo data vytvoření, a to buď sestupně, nebo vzestupně.



Obr. 23 Ukázka seznamu vytvořených formulářů

7.12 Zobrazení odpovědí na formulář

Jak již bylo zmíněno výše, může si uživatel zobrazit odpovědi na formulář dvěma způsoby – buď samostatné odpovědi na formulář, nebo odpovědi na individuální prvky formuláře. Zpracování odpovědí má na starost *ResponsePresenter*.

Při výběru možnosti pro zobrazení samostatných odpovědí se uživateli zobrazí v šabloně *showResponses* opět základní přehled v tabulce, kde vidí, kdy bylo na formulář odpovězeno a na jakou jazykovou verzi formuláře respondent odpověděl. Dále má možnost zobrazit danou odpověď na formulář, případně ji smazat.

Dole pod tabulkou se nachází tlačítko pro možnost exportu těchto odpovědí ve formátu CSV, kterou má na starost metoda `array2csv($data)`. Tato metoda je volána v *ResponsePresenteru* uvnitř metody `formSucceededExport($ids, $individual)`.

Samotný export může uživatel provést dvěma způsoby. Buď tak, že si vybere konkrétní odpovědi, které chce vyexportovat, nebo si může exportovat všech-

ny odpovědi hromadně. V případě individuálních odpovědí jsou id těchto odpovědí uloženy v proměnné `$ids` a v proměnné `$individual` je boolean proměnná 1.

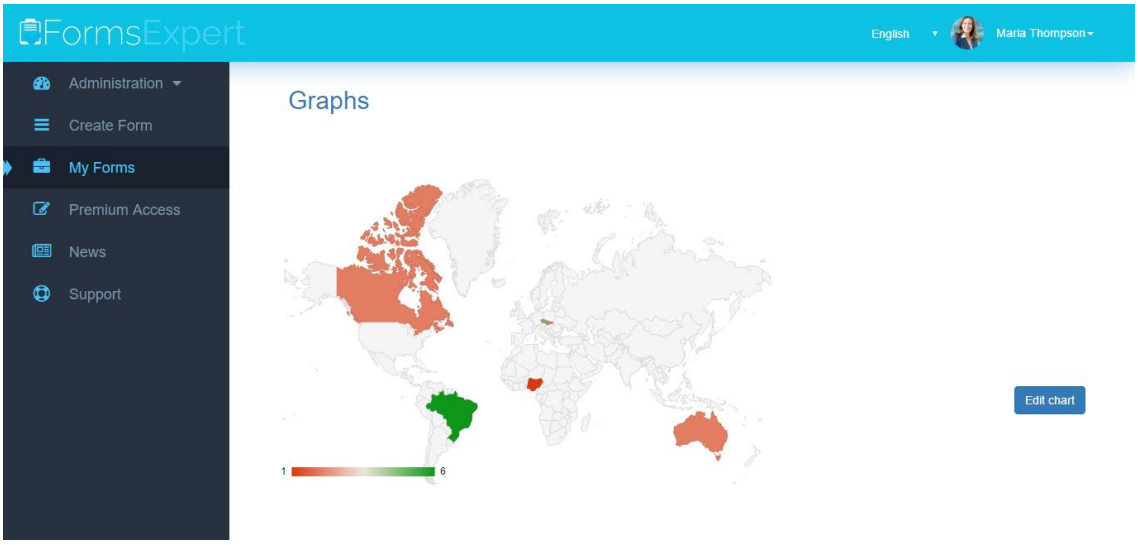
Jestliže si uživatel vybere možnost zobrazit pouze odpovědi na konkrétní prvek formuláře, zobrazí se mu tabulka s prvky daného formuláře a po výběru konkrétního prvku se mu vykreslí tabulka se všemi odpověďmi na něj. V této tabulce má vedle odpovědi možnost zobrazit odpovědi i na zbylé prvky ve formuláři.

7.13 Grafické vyhodnocení

Grafické vyhodnocení odpovědí slouží pro základní přehled o tom, jak respondenti odpovídali na konkrétní otázky. V případě, že odpovědi lze číselně vyhodnotit, což se týče například checkboxů, radio buttonů, select boxů, je vygenerován defaultně výsečový graf. Tento graf si následně uživatel může barevně upravovat, nastavovat mu vlastnosti a taktéž přepínat na jiné druhy grafů, například sloupcový. V případě select boxu s předdefinovanými zeměmi lze využít i zobrazení světa, kde jsou barevně (barva závisí na počtu odpovědí) vybarveny všechny státy, které respondenti vybrali jako odpověď.

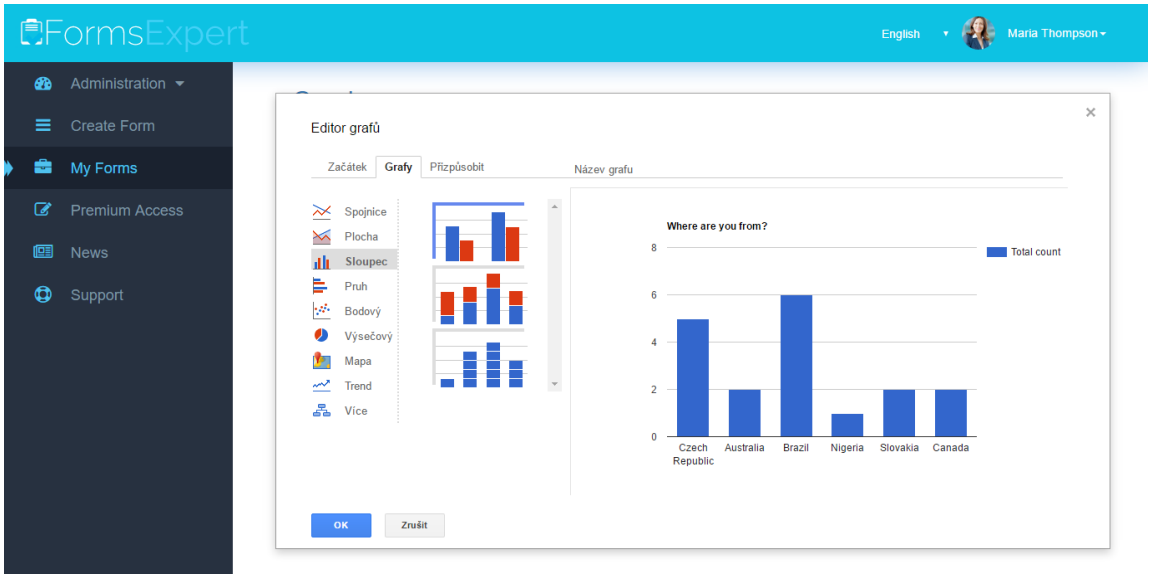
Pro vykreslování grafů je využito volně dostupných grafů Google Charts, jak již název vypovídá, od společnosti Google. Aby grafy mohly být vykresleny, je potřeba jednoduchého JavaScriptu, který se vloží do zdrojového kódu stránky. Jako první krok je nutno načíst Google Chart knihovny, dále načíst balíček pro námi zvolený graf, ve funkci `drawChart()` načíst data, která mají být v daném grafu interpretována, nastavit vlastnosti grafu a nakonec vytvořit objekt s id hodnotou, kterou si sami zvolíme. Poté už stačí na webové stránce vytvořit *div* element s tímto id, díky čemuž Google Chart do tohoto *div* elementu daný graf vykreslí.

V případě vykreslování více grafů, což je pro potřeby grafického vyobrazení odpovědí na více prvků formuláře nutné, je potřebné taktéž vytvořit více objektů. Těmto objektům, stejně tak i elementům *div*, jsou přiřazovány uvnitř funkce `drawChart()` unikátní názvy. K tomu je využito makra pro cyklus `{foreach}`, kde je k těmto názvům přiřazena unikátní hodnota id formulářového prvku.



Obr. 24 Ukázka grafického vyhodnocení odpovědí

Aby mohl uživatel měnit typ grafu podle svých potřeb, je nutné mít ve zdrojovém kódu stránky další JavaScript, který namísto balíčku *corechart* využívá balíček *chartEditor*. Dále je kód velmi podobný. Získání a zpracování odpovědí od respondentů, které jsou potřebné pro grafické zobrazení, má na starost funkce `getDataForVisualization()` v *ResponsePresenteru*. Ta odpovědi získává pomocí metody `getFormResponses()` v *MainModelu* a následně je uloží do pole tak, aby byly ve vhodném formátu pro Google Charts, a v metodě `renderGraphs()` jsou předány šabloně *graphs*.



Obr. 25 Ukázka editace grafu

7.14 Statistiky

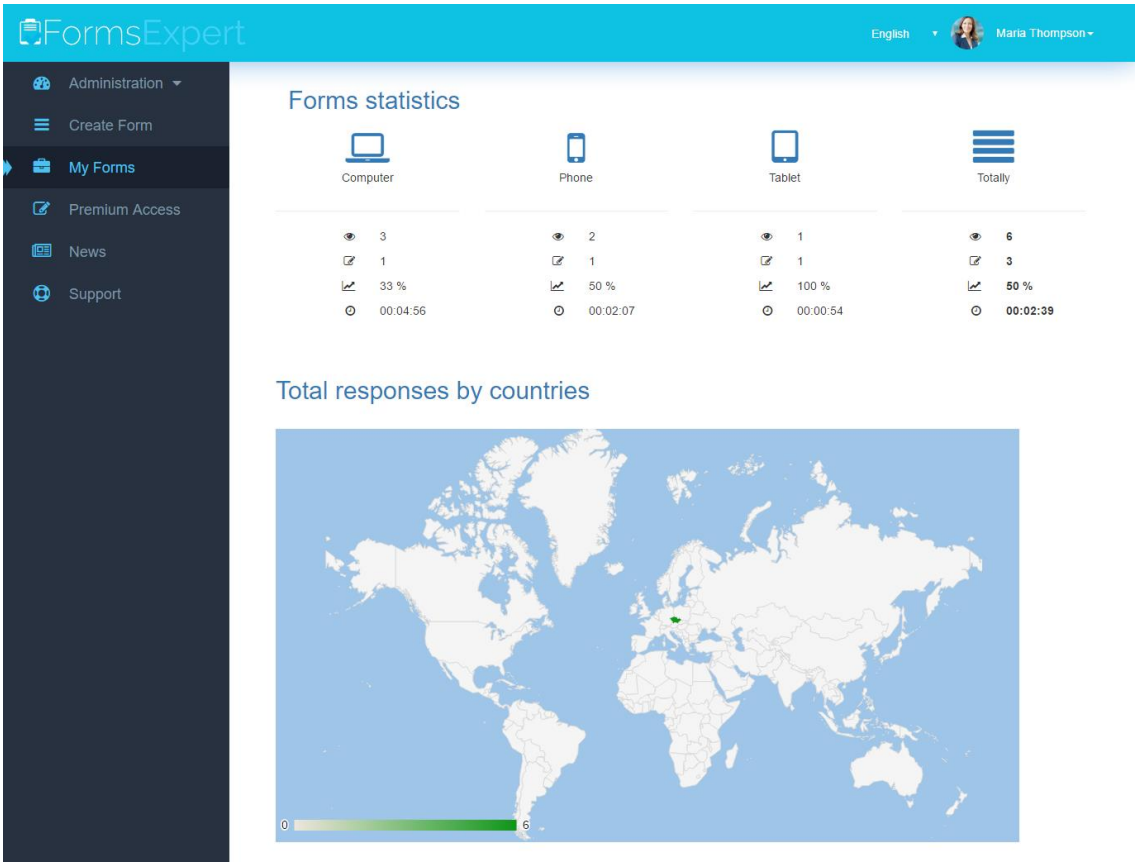
Poslední možností, kterou uživatel v základním přehledu vytvořených formulářů má, je zobrazení základních metrických statistik. V těchto statistikách je zachycen počet zobrazení formuláře, počet odpovědí na formulář, poměr mezi zobrazením a počtem odpovědí na formulář a také průměrná doba, která byla potřebná pro vyplnění formuláře. Tyto údaje se získávají z tabulky *respondent_info* a pomocí metody `getRespondentInfoData($formId)`, která se nachází ve *FormStatisticsPresenteru*, jejíž ukázka je níže, jsou tyto hodnoty rozděleny do sloupců podle zařízení, ze kterého bylo na daný formulář odpovídáno. Poslední sloupec zachycuje celkový souhrn ze všech zařízení dohromady.

```
$respondentInfoData = $this->main->getRespondentInfo($formId);

foreach ($respondentInfoData as $item) {
   //save totallyShowed value to each device
   $data[$item->device]['totallyShowed'] ++;
   $data['Totally']['totallyShowed'] ++;

        //save number of totally responded forms
        if ($item->finish_timestamp !== NULL) {
           $data['Totally']['totallyResponded'] ++;
           $data['Totally']['totalTime'] += $this->getDifferenceDatetime(
                $item->start_timestamp, $item->finish_timestamp);
           $data[$item->device]['totallyResponded'] ++;
           $data[$item->device]['totalTime'] += $this->getDifferenceDatetime(
                $item->start_timestamp, $item->finish_timestamp);
        }
}
```

Pod statistikami se navíc nachází mapa se základním přehledem o tom, ze kterých zemí daní respondenti odpovídali. K tomu se opět využívá graf od Google Charts.



Obr. 26 Ukázka statistického vyhodnocení odpovědí

7.15 Novinky

Administrátor může přidávat v *AdminModule* v presenteru *News* články, které se budou následně zobrazovat uživatelům. Uživatel je poté na nové články upozorněn tak, že se vedle položky News v levém menu nachází zvýrazněné číslo, které uvádí počet nových nepřečtených článků v novinkách. Z hlediska uživatelské přívětivosti je uživateli zobrazena hodnota pouze těch článků, které byly přidány za dobu od jeho posledního odhlášení, a to pouze za posledních 30 dní. Tím se předchází tomu, aby se mu zbytečně nezobrazoval vysoký počet nových příspěvků, a taktéž aby nebyl opakovaně upozorňován na nové příspěvky, které si během svého předposledního přihlášení mohl přečíst, ale neudělal to.

Zobrazení hodnoty je řešeno pomocí metody `startup()` v *BasePresenteru*, kdy je pokaždé šabloně předána hodnota `$unreadedNews`, která je ve skutečnosti velikostí pole, které vrací metoda v *MainModelu* `getNewNews($userId)`. Ta porovná nově přidané články v novinkách za dobu od posledního odhlášení uživatele (případně za posledních 30 dní) s hodnotami v tabulce *readed_news*, kde jsou uloženy uživatelem přečtené články, a vrátí pole s id článků, které v tomto časovém rozmezí nebyly přečteny.

```
$this->template->unreadedNews = sizeof($this->main
      ->getNewNews($this->getUser()->getId()));
```

Pokud si uživatel chce přečíst novinky, je mu zobrazen pomocí metody `getNews($newsId)` v šabloně *default* seznam několika posledních příspěvků, kde je ilustrační obrázek, titulek a první odstavec textu. Tato metoda může zobrazit poslední novinky (v případě, že proměnná $`newsId` je *NULL*), případně konkrétní článek v novinkách (pokud je v `$newsId` jeho id).

```
public function getNews($newsId = NULL) {
    if ($newsId === NULL) {
        return $this->selectTable('news')
                    ->order('date DESC')
                    ->limit(10);
    } else {
        return $this->selectTable('news')
                    ->where('id = ?', $newsId);
    }
}
```

Opět se využívá pole `$unreadedNews`, díky kterému lze ke každému článku přidat zvýrazněné upozornění, že se jedná o nový, respektive nepřečtený, článek. Po výběru konkrétního článku je přesměrován na šablonu *show*, kde je již zobrazen celý článek a má zde možnost přidávat komentáře.

7.16 Komentáře

Jak bylo zmíněno výše, může každý uživatel pod dané články v novinkách přidávat komentáře. Jedná se v podstatě o jednoduché komentování článků, kdy jsou komentáře zobrazovány pod sebou bez možnosti reagovat na konkrétní komentář. Metoda `getNewsComments($newsId)` má na starost získání všech komentářů k danému článku. Ty jsou následně pod článkem vykresleny pomocí makra `{foreach}`. Pro odeslání nového komentáře lze využít formuláře, který je vytvořen metodou `createComponentCommentForm()`.

Autor komentáře má samozřejmě možnost svůj vlastní komentář smazat, pokud v něm udělal chybu nebo z jakéhokoliv jiného rozhodnutí, jenž ho k tomuto kroku vedlo. K tomu slouží tlačítko *delete* na pravé straně komentáře, které volá akci `actionDeleteComment($commentId)`. Komentáře může samozřejmě mazat i administrátor, pokud zjistí, že některý z komentářů ze zjevného důvodu do dané diskuse nepatří.

7.17 Prémiový přístup

V aplikaci je již naimplementovaná možnost platby přes PayPal, díky které v budoucnu budou moci uživatelé platit za možnost vytvářet formuláře. V současné době je její provoz pouze ve fázi testovacího režimu, neboť prozatím není stanovena politika, jakým způsobem bude využívání aplikace zpoplatněno.

Pomocí Composeru bylo nutné stáhnout rozšíření *seberm/paypal-component* a následně v *config.neon* nakonfigurovat a zaregistrovat toto rozšíření. V konfiguraci je nutné nastavit jméno, heslo a vygenerovaný podpis, které lze najít v nastavení svého účtu na stránkách PayPal. Jelikož se jedná prozatím o testovací režim, je nutné mít v parametru *sandbox* nastavenou hodnotu *true*.

Metody, které zajišťují provedení platby, se nachází v *SimplePayment* presenteru v *UserModule*.

- `createComponentMonthsAccessForm()` – slouží k vykreslení select boxu, který obsahuje hodnoty získané z tabulky *access_types*. Uživatel si vybere některou z možností a na základě odeslání tohoto formuláře se volá následující metoda.
- `formSucceeded($form, $values)` – tato metoda slouží k přesměrování na šablonu *paypal* spolu s parametrem v URL adrese, díky němuž se může komponenta PaypalButton vykreslit se správnými údaji o této platbě.
- `createComponentPaypalButton()` – touto metodou je vykresleno tlačítko PayPal, které má za úkol přesměrovat uživatele na stránky PayPal, kde se mu zobrazí formulář (nákupní karta) s informacemi o produktu, v tomto případě službě, která je nakupována. Tyto hodnoty (název produktu, cena, počet kusů, daň) jsou předány metodě `addItemToCard()`, která se nachází v API rozšíření PayPal a má za úkol předat tyto hodnoty PayPalu. V této komponentě mohou být volány celkem tři handlery událostí – `onConfirmation[]`, `onError[]`, `onCancel[]`.
- `confirmOrder($data)` – tato metoda je volána v případě úspěšné platby. Volá ji handler událostí `onConfirmation[]`. Je volána ve chvíli potvrzení objednávky uživatelem. V případě, že platba proběhla v pořádku, je volána metoda níže.
- `successPayment($data)` – zde dochází k uložení provedené platby do databáze do tabulky *access*. V proměnné `$data` jsou předány informace o dané objednávce, které je potřeba do této tabulky uložit.
- `errorOccurred($errors)` – v případě, že platba z nějakého důvodu neproběhla, je handlerem událostí `onError[]` volána tato metoda, která má na starost vykreslit chybovou hlášku o neproběhlé platbě a přesměrovat uživatele opět na stránku s výběrem platby.
- `canceled($data)` – pokud se uživatel před samotnou platbou rozhodne zrušit objednávku kliknutím na tlačítko *Cancel*, je zavolána handlerem `onCancel[]` tato metoda, která má stejně jako metoda výše za úkol vy-

kreslit uživateli hlášku o zrušení platby a přesměrovat jej na stránku s výběrem platby.

7.18 Administrace

Pokud má uživatel administrátorská práva, je mu v levém menu zobrazena rozbalovací nabídka Administration, díky níž může spravovat registrované uživatele, přidávat nové uživatelské recenze, vkládat nové příspěvky do novinek, získat základní přehled o uživatelích a platbách a taktéž vytvářet a upravovat domény, na nichž může daná aplikace běžet. Všechny presentery, které budou zmiňovány, se nachází v *AdminModulu*. V tomto modulu je v *BasePresenteru* v metodě `startup()` kontrolováno oprávnění uživatele, zda vůbec může k daným presenterům a jejich metodám přistupovat. V případě, že nemá dostatečná oprávnění, je přesměrován na *HomepagePresenter* v *UserModulu*.

7.19 Správa uživatelů

Pro vykreslení všech uživatelů se využívá v šabloně *default* v *UsersPresenteru* opět TwiGrid tabulka se základním přehledem o uživatelích – jejich email, jméno, jazyk, roli. Dále se zde nachází možnosti na další práci s uživatelem – historii o platbách, registrační údaje a taktéž lze zobrazit jeho vytvořené formuláře. V této tabulce lze přímo uživateli upravovat jejich základní údaje, např. email, role, případně jej jedním kliknutím okamžitě deaktivovat, aby se již nemohl přihlásit. Nechybí zde ani filtry pro vyhledávání konkrétního uživatele na základě emailu, jména a podobně.

V případě, kdy administrátor vybere možnost zobrazení platební historie uživatele, je přesměrován na šablonu *payment* v *UsersPresenteru*, kde je vykreslena tabulka s historií plateb daného uživatele a zároveň možnost danému uživateli vytvořit prémiový přístup, což má na starost metoda `createComponentAccessForm()`, která vytvoří jednoduchý formulář, obsahující select box s výběrem konkrétního prémiového přístupu. Po odeslání tohoto formuláře je zavolána metoda `formSucceeded($form, $values)`, která se postará o uložení dat do databázové tabulky *access*.

Pokud si chce administrátor zobrazit formuláře daného uživatele, je přesměrován na šablonu *forms* v *UserPresenteru* v *UserModule* spolu s parametrem obsahujícím id daného uživatele. Díky tomu může vidět tabulku s formuláři tak, jak by ji viděl daný uživatel, avšak kvůli možným citlivým údajům nemůže ani s právy administrátora zobrazit konkrétní odpovědi na uživatelovy formuláře.

7.20 Přidání recenze

Na hlavní straně webové aplikace se nachází uživatelské recenze. Tyto recenze může vkládat pouze administrátor na základě hodnocení uživatelů, které byly

zaslány na email. K tomu slouží formulář vytvořený metodou `createComponentAddReview()`. Do formuláře vloží jméno, fotografii, profesi a samotný text uživatele, který recenzi zaslal. Po odeslání formuláře je volána metoda `formAddReviewSucceeded($form, $values)`, která recenzi uloží do tabulky *reviews*.

Na stejné stránce si může vybrat ještě možnost zobrazení TwiGrid tabulky s již existujícími recenzemi, které může následně upravovat, podobně jako u předchozích TwiGrid tabulek.

Samotné zobrazení existujících recenzí má na starost metoda `getRandomReviews($number)` ve *FrontModulu* v *MainpagePresenteru*, která vybere náhodně určitý počet recenzí. Tento počet určuje proměnná `$number`. Jelikož se na stránce v jednu chvíli zobrazují právě dvě recenze, je v této metodě kontrolováno, zda je hodnota v proměnné `$number` sudá. Pokud ne, je tato proměnná snížena o jedna. Tyto recenze se následně zobrazují v šabloně *default*.

7.21 Přidání novinek

Obdobně jako přidávání uživatelských recenzí, lze přidávat příspěvky do novinek. O to se stará metoda na vykreslení formuláře `createComponentAddNews()` v presenteru *News*. Do formuláře vloží titulek, obrázek a tag, reprezentující daný příspěvek. Na rozdíl od recenzí je text, který bude tvořit samotný příspěvek, možno formátovat pomocí WYSIWYG TinyMCE. Dané příspěvky lze samozřejmě případně editovat a mazat. To mají na starosti metody `actionEditNews($newsId)` a `actionDeleteNews($newsId)`.

7.22 Přehled plateb

Přehled plateb zajišťuje *PaymentPresenter*. Platby lze zobrazit dvěma způsoby. Buď tabulkou TwiGrid, kde lze procházet poslední platby s informacemi o platících uživatelích, nebo graficky pomocí Google Charts grafů. Zde se nachází dva grafy – jeden obsahující celkovou sumu, která byla uživateli zaplacena, druhý celkový počet druhů, které si uživatelé kupovali. Oba grafy lze filtrovat na základě časového intervalu, navíc lze nastavit, zda se mají tyto hodnoty seskupovat na základě dnů, měsíců nebo let. O to se stará metoda `getPaymentData($fromDate, $toDate, $sortingFormat)`. Proměnné jsou v této metodě defaultně NULL, tudíž jsou grafy zobrazeny bez formátování. Avšak ve chvíli, kdy je odeslán formulář nacházející se nad těmito grafy, jenž je vytvořený metodou `createComponentPaymentReportFormatingForm()`, je tato metoda opětovně volána metodou `formSucceeded($form, $values)`, kdeproměnná `$values` již obsahuje formátovací hodnoty.

7.23 Vícejazyčnost webové aplikace

Jelikož aplikaci by měli alespoň z počátku využívat především anglicky mluvící zákazníci, je primárně celá aplikace v angličtině. Avšak základní prvky aplikace, jako je menu a podobně, lze přepínat do více jazyků – prozatím čeština, slovenština a němčina.

Aby mohl být web přeložen do více jazyků, bylo potřeba stáhnout rozšíření *kdyby/translator* pomocí Composeru, a do hlavního *BasePresenteru* ve složce presenters, ze kterého následně dědí ostatní *BasePresentery* v modulech, bylo nutné přidat persistentní proměnou `$locale` a injectovat službu *Translator*, která má na starost samotný překlad.

Dalším krokem bylo potřeba upravit router tak, aby obsahoval parametr *locale* s konkrétním jazykem. Díky tomu bude URL adresa vždy obsahovat tento parametr, na jehož základě bude překladač vědět, který jazyk má být použit.

```
$router[] =
      new Route('[<locale=en en|de|cs|sk>/]<presenter>/<act.ion>',
      "Homepage:default");
```

Na závěr již stačilo do adresáře *app/lang* přidat slovníkové soubory. Pro češtinu konkrétně *ui.cs_CZ.neon*, kde *ui* je pojmenování slovníku a *cs_CZ* je kombinace jazyka a státu (podle ISO639-1 a ISO 3166–1). U ostatních tří slovníků je pojmenování obdobné.

Lokalizace webu je závislá především na defaultním jazyce, který si uživatel zvolí při registraci. Avšak v případě registrace a přihlašování není tato hodnota známa, proto se jazyk webu řídí většinou z informace o jazyce prohlížeče, která je uložena v http hlavičce. Pokud informaci o jazyce nezíská ani tímto způsobem, je použit defaultní jazyk, který je uveden v nastavení v souboru *config.neon*, což je v tomto případě angličtina.

7.24 Přidání domén

Z důvodu informování budoucích zákazníků byl vytvořen informativní *ComingsoonPresenter*, který se nachází ve *FrontModulu*. Ten měl za úkol zobrazovat odpočet s informací, za jakou dobu bude dostupná betaverze této webové aplikace.

Avšak aby bylo možné k aplikaci přistupovat z určitých domén, tedy v případě vývoje z *localhostu*, nebo z domény společnosti, ve které byla aplikace vyvíjena, aniž by se zobrazoval výše uváděný odpočet, bylo potřeba vytvořit jednoduché rozhraní v *AdminModulu*, které do databáze uloží požadovanou doménu, ze které je možné k aplikaci přistupovat. Na základě takto uložených domén se v *RouterFactory* ověřuje, z jaké domény je k této aplikaci možno přistupovat.

Ke zjištění, z jaké domény se k aplikaci přistupuje, slouží následující řádek kódu přímo v *RouterFactory*. Zde je do proměnné `$domain` uložena boolean hodnota z volané metody `getDomains()` v *MainModelu*, která zjišťuje, zda se daná doména nachází v databázi v tabulce *Domains*.

```
$domain = $this->main->getDomains($_SERVER['SERVER_NAME']);
```

Jestliže se nenachází, je dostupný pouze *FrontModul* s *Comingsoon* Presenterem a k ostatním modulům a presenterům nelze přistupovat. Avšak pokud se v této tabulce doména nachází, lze přistupovat do celé aplikace. Obdobným způsobem lze nastavovat, zda lze zapnout nebo vypnout *ProductionMode*, případně *DebugMode*. Zjednodušená ukázka implementace je následující:

```
if ($domain) {
   $router[] = $commingSoonRouter = new RouteList('Front');
   $commingSoonRouter[] = new Route
        ($mainDomain.'/<presenter>/<action>[/<id>]',
        'Commingsoon:default');
} else {
    //routing Front/User/AdminModulu
}
```

Přidávání a odebírání domén se nachází v *AdminModulu* v *DomainsPresenteru*. Existující domény lze opět zobrazit v TwiGrid tabulce, kde je lze jednoduše jedním kliknutím aktivovat, případně deaktivovat. Zároveň zde lze přidat novou doménu. Přidání nové domény má na starost metoda `createComponentDomainForm()`, která vytvoří formulář s pouze jedním textovým polem a tlačítkem na odeslání. Po odeslání formuláře je volána metoda `formDomainsSucceeded($form, $values)`, která má na starost kontrolovat, zda již stejná doména není v databázi. Pro tuto kontrolu volá metodu `isDuplicatedDomain($values->domain)`. Pokud doména není duplicitní, uloží se do databáze. Kontrola duplicitních dat je důležitá zejména z toho důvodu, aby se do databáze neuložila aktivní doména, když už v databázi existuje stejná, která by byla například již dříve deaktivována.

8 Technické a ekonomické vyhodnocení

Ekonomické vyhodnocení vyvíjené aplikace je v současné chvíli poměrně složité, a to jak z pohledu vyčíslení nákladů, tak z pohledu budoucích výnosů. Co se týče nákladů, vývoj nebyl doposud dokončen a ani po úspěšném dokončení aplikace se vývoj zcela nezastaví. Z pohledu výnosů není prozatím stanovena platební politika, která by určila, jakým způsobem bude využívání formulářů zpoplatněno.

Z technického hlediska je aplikace ve své podstatě plně funkční. Uživatelé mohou vytvářet a rozesílat formuláře, zobrazovat a vyhodnocovat odpovědi. Aplikace sice obsahuje určité kosmetické nedostatky, které ale během následujícího vývoje budou eliminovány a nejsou technicky omezující pro zákazníka, využívajícího aplikaci. V podstatě by již stačilo stanovit platební politiku a aplikace by mohla přinášet určité výnosy.

Proto v této kapitole bude ekonomické vyhodnocení vycházet z nákladů na vývoj a samotný provoz aplikace, které byly vynaloženy do této chvíle. Výnosy budou odhadovány na základě pesimistického pohledu, tedy z nejnižší z cen konkurence, která byla analyzována v kapitole 5. Jelikož vývoj bude i nadále pokračovat a budou se implementovat další nové funkce, není rozumné okamžitě požadovat příliš vysokou částku. V prvé řadě je potřeba získat ze začátku alespoň část platících zákazníků, kteří by aplikaci využívali a ocenili by právě nižší cenu, za kterou ji mohou využívat. Z tohoto důvodu není ve výpočtu zohledněna možnost vyšší měsíční platby.

Nejnižší částku za měsíc využívání služeb najdeme u společnosti WuFoo, která kromě formulářů zdarma nabízí nejlevnější placenou variantu za 15 dolarů. To je v přepočtu 389 Kč. Jelikož do této chvíle byly přibližné náklady na vývoj ve výši 45 000 Kč, náklady na doménu a webhosting ve výši 1 600 Kč, tak by se návratnost investice projevila ve chvíli, kdy by aplikaci zaplatilo alespoň 120 zákazníků.

V tomto ekonomickém vyhodnocení je vycházeno z předpokladu, že ne všichni zákazníci potřebují vytvářet formuláře pravidelně, ale většina je využije pouze v krátkém období, tedy v rámci jednoho měsíce. To se týče 80 % zákazníků, zbylých 20 % bude využívat našich služeb pravidelně.

Z počátku není předpokládáno, že by byl velký zájem o využívání aplikace, takže reálně by mohlo být v prvním měsíci 10 platících zákazníků, jejichž počet by v následujících měsících vzrůstal o 25 %, v lepším případě až o 50 %. Avšak počet zákazníků lze navýšit vhodně cílenou reklamou, do které by bylo nutné investovat. Investice ve výši 40 000 Kč by mohla odhadem přinést třikrát více zákazníků oproti situaci, kdy by žádná investice do reklamy nebyla.

Z těchto předpokladů tedy vychází (viz přílohy B a C), že v případě, kdy by nebylo investováno do reklamy, vrátila by se investice v obou případech přírůstku zákazníků během 5 měsíců. Po půl roce by byl v případě 25% přírůstku výnos téměř 22 000 Kč, v případě 50% přírůstku již téměř 68 000 Kč.

Pokud by však byly náklady zvýšené o investici do reklamy, ziskovost aplikace by byla v obou případech již ve 4. měsíci, ale výnosy by po půl roce byly několikanásobně vyšší – v případě nižšího přírůstku zákazníků by výnos činil 118 000 Kč a v případě vyššího přírůstku až 256 000 Kč.

Jak je vidět, vyplatí se investovat do reklamy a získat více zákazníků, neboť i v horším případě, kdy by se vyplnil odhad nižšího přírůstku zákazníků, by po půl roce fungování aplikace v plném provozu mohl být výnos dvaapůlkrát vyšší, než jaké byly náklady na vývoj.

V tomto ekonomickém vyhodnocení je počítáno pouze to, za jak dlouho se navrátí investice do nynější verze aplikace, pokud by již dále nebyla vyvíjena. Jelikož však vývoj bude i nadále pokračovat, budou samozřejmě růst i náklady. Tyto náklady mohou být časem zvýšeny i z důvodu přijetí nových vývojářů, díky kterým by vývoj mohl být urychlen a zefektivněn. S rostoucí funkcionalitou a dalšími možnostmi aplikace by mohl růst i počet spokojených zákazníků, a to možná i rychleji, než je odhadováno v tomto ukázkovém příkladu. S rostoucí funkcionalitou by tedy mohla postupem času vzrůst i cena, kterou by zákazníci byli ochotni platit, což by vedlo opět ke zvyšování výnosů, které by tyto náklady pokrývaly. Případně by se mohly v budoucnu zavést placené varianty od nejlevnější až po nejdražší, kde by cena odpovídala možnostem práce s formuláři, jaké by uživatel mohl v aplikaci využívat.

Z odhadu vývoje návratnosti investice se ale dá očekávat, že i při budoucích, v současné době těžce odhadnutelných, nákladech na vývoj je velmi pravděpodobná ziskovost.

9 Závěr a diskuze

Cílem diplomové práce bylo navrhnout a implementovat webovou aplikaci na tvorbu formulářů.

V kapitole o PHP frameworcích byly analyzovány v současnosti nejpoužívanější PHP frameworky, které jsou mezi vývojáři nejoblíbenější. Ve světě jsou to především Laravel, Symfony, CodeIgniter, Yii a dále i v České republice vyvíjený framework Nette, v němž probíhal samotný vývoj webové aplikace.

V kapitole o webdesignu byl popsán současný trend responzivního webového designu, především Bootstrap, dále i JavaScript a knihovna jQuery, které jsou při vývoji webů taktéž nejčastěji používány.

Kapitola Analýza konkurenčních řešení se zabývala pohledem na některé ze známých konkurenčních řešení tvorby formulářů. Na základě této analýzy bylo provedeno vyhodnocení daných konkurenčních řešení, především jejich klady a zápory, na jejichž základě byla vytvořena specifikace na funkcionalitu vlastní aplikace.

Následně byl v kapitole Návrh aplikace proveden návrh samotné webové aplikace. Prvně byly pomocí Use Case diagramů provedeny případy použití, dále byl vytvořen grafický návrh celé aplikace a navržena databáze. Na základě návrhu proběhla implementace samotné aplikace.

Aplikace zvládá uživatelsky jednoduchou a intuitivní tvorbu formulářů s mnoha nastaveními jeho prvků a vlastností. Kromě správy formulářů a jejich rozesílání respondentům je k dispozici přehledné zobrazování jejich odpovědí, a to nejen v textové podobě, ale i v podobě grafické, spolu se základními metrickými statistikami.

V aplikaci je dostupné i rozhraní pro administraci, v němž má administrátor základní přehled o všech uživatelích, jejich platbách a formulářích. Jelikož je aplikace modulární, lze do budoucna její rozhraní dále rozšiřovat a přidávat další role. S ohledem na uvedený předpokládaný cíl práce lze tedy konstatovat, že byl tento cíl splněn a aplikace je úspěšně naimplementována.

Přestože je aplikace z hlediska funkcionality již připravena k produkčnímu nasazení, vývoj tím v žádném případě nekončí. Aby byla aplikace i do budoucna konkurenceschopná, musí se neustále vyvíjet. Proto se v blízké době dále naimplementují další prvky, které lze přidat do formuláře ve FormBuilderu, a taktéž se naimplementuje logické skrývání a zobrazování prvků formuláře na základě odpovědí respondenta.

Dále bude naimplementováno základní zabezpečení formulářů proti vícenásobným odpovědím robotů a uživatelů. V případě robotů by pravděpodobně nejlepším řešením byla ochrana pomocí Captcha, která by musela být potvrzena nebo vyplněna před samotným odesláním formuláře. Kromě ošetření vícenásobných odpovědí robotů je potřeba ošetřit i vícenásobné odpovědi uživatelů. Proto jsou v tabulce *Respondent_info* uloženy základní veřejné informace o respondentovi, díky kterým by se dalo relativně jednoduše porovnat, zda byly odpovědi odeslány vícekrát z jednoho počítače.

Rovněž vznikne další modul, který bude spravovat uživatel s rolí moderátora, jenž bude oprávněn řešit se zákazníky jejich problémy pomocí chatu. V nejbližší době bude vyřešena platební politika za využívání služeb, díky čemuž bude aplikace přinášet zisk, který pokryje všechny náklady vzniklé během minulého i budoucího vývoje.

10 Literatura

BEAN, MARTIN. *Laravel 5 Essentials: Explore the fundamentals of Laravel, one of the most expressive and robust PHP frameworks available*. Birmingham: Packt Publishing Ltd., 2015. ISBN 978-1-78528-301-7.

Bootstrap: Getting started [online]. [cit. 2016-11-08]. Dostupné z: http://getbootstrap.com/getting-started/

CodeIgniter Web Framework: Why CodeIgniter? [online]. [cit. 2016-11-08]. Dostupné z: https://www.codeigniter.com/

CRANLEY, R. -- BENEDETTI, R. *Head first jQuery*. *Sebastopol,* CA: O'Reilly Media, 2011. 544 s. ISBN 978-1-4493-9321-2.

Doctrine Project: Object Relational Mapper [online]. [cit. 2016-11-08]. Dostupné z: http://www.doctrine-project.org/projects/orm.html

Formstack: Online Form Builder Solution [online]. [cit. 2016-09-15]. Dostupné z: https://www.formstack.com

Formuláře Google [online]. [cit. 2016-09-15]. Dostupné z: https://docs.google.com/forms/u/0/

GitHub: dobtco/formbuilder [online]. 2014 [cit. 2016-12-25]. Dostupné z: https://github.com/dobtco/formbuilder

GUAY, MATTHEW. *The 14 Best Online Form Builders for Every Task* [online]. 2016 [cit. 2016-09-17]. Dostupné z: https://zapier.com/learn/ultimate-guide-to-forms-and-surveys/best-online-form-builder-software/

JotForm: The Easiest Online Form Builder [online]. [cit. 2016-09-15]. Dostupné z: https://www.jotformeu.com

Laravel - The PHP Framework For Web Artisans [online]. [cit. 2016-11-08]. Dostupné z: https://laravel.com/docs/5.3

MARCOTTE, ETHAN. *An A List Apart Article: Responsive Web Design* [online]. 2010 [cit. 2016-11-08]. Dostupné z: http://alistapart.com/article/responsive-web-design

Nette framework: Rychlý a pohodlný vývoj webových aplikací v PHP [online]. [cit. 2016-11-08]. Dostupné z: https://nette.org/

Noetiforce: PHP Frameworks: The Best 10 for Modern Web Development [online]. 2016 [cit. 2016-11-08]. Dostupné z: http://noeticforce.com/best-php-frameworks-for-modern-web-development

PETERSON, CLARISSA. *Learning responsive Web design: a beginner's guide. Sebastopol, CA: O'Reilly Media, Inc., 2014. ISBN 144936294X.*

PITT, CHRIS. *Pro PHP MVC. New York: Distributed to the book trade worldwide by Springer Science+Business Media, c2012. Expert's voice in open source. ISBN 9781430241652.*

RIVOAL, FLORIAN (ed.). *CSS Working Group Editor Drafts: Media Queries Level 3* [online]. 2012 [cit. 2016-11-08]. Dostupné z: https://drafts.csswg.org/mediaqueries-3/#width

SKVORC, BRUNO. *SitePoint: The Best PHP Framework for 2015: SitePoint Survey Results* [online]. 2015 [cit. 2016-11-08]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

SMITH, BRIALLYN. *The Finest 14 Google Forms Alternatives You Should Try* [online]. 2016 [cit. 2016-09-15]. Dostupné z: http://www.makeuseof.com/tag/finest-14-google-forms-alternatives-try/

Symfony: Symfony 3.1 Documentation [online]. [cit. 2016-11-08]. Dostupné z: https://symfony.com/doc/

The Tech Terms Computer Dictionary: Framework Definition [online]. 2013 [cit. 2016-11-08]. Dostupné z: http://techterms.com/definition/framework

Tutorial Republic: Working with Bootstrap 3 Responsive Layout [online]. [cit. 2016-11-08]. Dostupné z: http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-responsive-layout.php

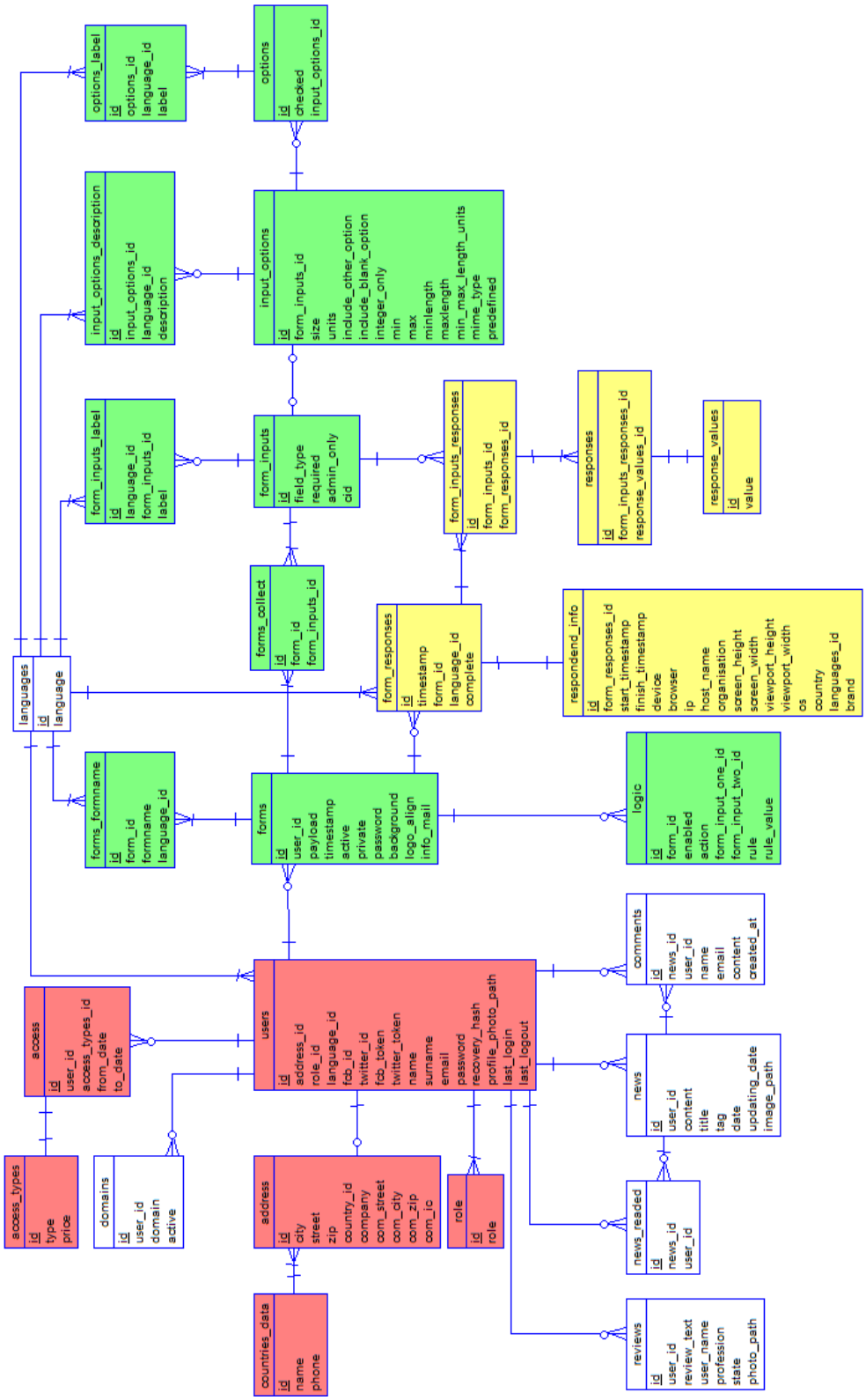
Typeform: Free & Beautifully Human Online Forms [online]. [cit. 2016-09-15]. Dostupné z: http://www.typeform.com/

Wufoo: Online Form Builder with Cloud Storage Database [online]. [cit. 2016-09-15]. Dostupné z: http://www.wufoo.com/

ZAKAS, N. *Professional JavaScript for web developers*. Indianapolis: Wiley, 2012. 960 s. ISBN 978-1-118-02669-4.

Přílohy

A ERD návrh databáze



B Ekonomické vyhodnocení bez reklamy

Vývoj	46 600 Kč
Reklama	0 Kč
Celková investice	46 600 Kč

25% přírůstek zákazníků měsíčně						
Měsíc	Noví zákaznící	Zákazníci, kteří zůstali	Celkem věrných zákazníků	Celkem plateb	Celkem Kč	Návratnost investice
1	10			10	3 890 Kč	- 42 710 Kč
2	13	2	2	27	10 503 Kč	- 36 097 Kč
3	16	3	5	51	19 839 Kč	- 26 761 Kč
4	20	3	8	82	31 898 Kč	- 14 702 Kč
5	25	4	12	123	47 847 Kč	**1 247 Kč**
6	31	5	17	176	68 464 Kč	**21 864 Kč**
7	39	6	23	244	94 916 Kč	**48 316 Kč**

50% přírůstek zákazníků měsíčně						
Měsíc	Noví zákaznící	Zákazníci, kteří zůstali	Celkem věrných zákazníků	Celkem plateb	Celkem Kč	Návratnost investice
1	10			10	3 890 Kč	- 42 710 Kč
2	15	2	2	29	11 281 Kč	- 35 319 Kč
3	22,5	3	5	60	23 340 Kč	- 23 260 Kč
4	33,75	5	10	109	42 401 Kč	- 4 199 Kč
5	50,625	7	17	184	71 576 Kč	**24 976 Kč**
6	75,9375	10	27	297	115 533 Kč	**68 933 Kč**
7	113,9063	15	42	468	182 052 Kč	**135 452 Kč**

C Ekonomické vyhodnocení s reklamou

Vývoj	46 600 Kč
Reklama	40 000 Kč
Celková investice	86 600 Kč

25% přírůstek zákazníků měsíčně						
Měsíc	Noví zákaznící	Zákazníci, kteří zůstali	Celkem věrných zákazníků	Celkem plateb	Celkem Kč	Návratnost investice
1	30			30	11 670 Kč	- 74 930 Kč
2	39	6	6	81	31 509 Kč	- 55 091 Kč
3	48	8	14	151	58 739 Kč	- 27 861 Kč
4	60	10	24	245	95 305 Kč	**8 705 Kč**
5	75	12	36	368	143 152 Kč	**56 552 Kč**
6	93	15	51	527	205 003 Kč	**118 403 Kč**
7	117	19	70	733	285 137 Kč	**198 537 Kč**

50% přírůstek zákazníků měsíčně						
Měsíc	Noví zákaznící	Zákazníci, kteří zůstali	Celkem věrných zákazníků	Celkem plateb	Celkem Kč	Návratnost investice
1	30			30	11 670 Kč	- 74 930 Kč
2	45	6	6	87	33 843 Kč	- 52 757 Kč
3	67,5	9	15	179	69 631 Kč	- 16 969 Kč
4	101,25	14	29	323	125 647 Kč	**39 047 Kč**
5	151,875	20	49	544	211 616 Kč	**125 016 Kč**
6	227,8125	30	79	881	342 709 Kč	**256 109 Kč**
7	341,7188	46	125	1394	542 266 Kč	**455 666 Kč**