

Návrh a realizace webové aplikace pro zjednodušení uzavírání pojištění přes internet.

Bc. Filip Hrubý

Diplomová práce
2015/16



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2015/2016

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Filip HRUBÝ**
Osobní číslo: **A14465**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Návrh a realizace webové aplikace pro zjednodušení uzavírání pojištění přes internet**

Téma anglicky: **The Design and Implementation of a Web Application that Simplifies the Conclusion of Insurance Policies over the Internet**

Zásady pro vypracování:

1. Vypracujte stručný rozbor technologií, které budou použity k vývoji aplikace.
2. Provedte analýzu požadavků a uživatelských cílů na zvolené řešení.
3. Navrhněte vhodné řešení aplikace.
4. Realizujte funkční prototyp navržené aplikace.
5. Věnujte pozornost zabezpečení aplikace.
6. Popište nasazení vytvořené aplikace a prostředky nutné pro běh.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. BEAN, Martin. Laravel 5 Essentials. Vyd. 1. UK: PACKT, 2015. ISBN 9781785283017.
2. GUTMANS, Andi, Stig S?ther BAKKEN a Derick RETHANS. Mistrovství v PHP 5. Vyd. 1. Brno: CP Books, 2005, 655 s. ISBN 80-251-0799-x.
3. DAIGNEAU, Robert. Service design patterns: fundamental design solutions for SOAP/WSDL and RESTful Web services. Upper Saddle River, NJ: Addison-Wesley, 2012, xxv, 321 p. Addison-Wesley signature series. ISBN 032154420x.
4. STRAUB, Ben. Pro Git. Vyd. 2. UK: APRESS, 2014. ISBN 9781484200773.
5. SOMMERVILLE, Ian. Softwarové inženýrství. 1. vyd. Brno: Computer Press, 2013, 680 s. ISBN 978-80-251-3826-7.
6. OCTOBER CMS documentation. octobercms [online] 2015 [cit. 2016-01-24]. Dostupné z: <https://octobercms.com/docs/setup/installation>
7. PERCONA documentation. octobercms [online] 2015 [cit. 2016-01-24]. Dostupné z: <https://www.percona.com/software/documentation>

Vedoucí diplomové práce:

Ing. Petr Šilhavý, Ph.D.

Ústav počítačových a komunikačních systémů

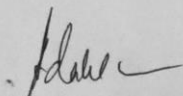
Datum zadání diplomové práce:

5. února 2016

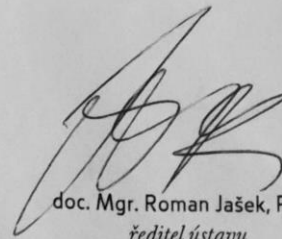
Termín odevzdání diplomové práce:

20. května 2016

Ve Zlíně dne 5. února 2016



doc. Mgr. Milan Adámek, Ph.D.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne



.....
podpis diplomanta

ABSTRAKT

Táto diplomová práca sa zaoberá aplikáciou pre zjednodušenie uzatvárania poistenia cez internet. V teoretickej časti sú popísané technológie ktoré boli využité pri vývoji. V praktickej časti sú zhrnuté všetky funkčné a nefunkčné požiadavky a na ich základe je popísaná funkčná špecifikácia. Záver praktickej časti je zameraný na implementáciu dvoch hlavných častí aplikácie, kde sú popísané jej hlavné body.

Klíčová slova: Webová aplikácia, Online poistenie, PHP, Laravel

ABSTRACT

This dissertation describes application for simplified conclusion of insurance policies through the internet. The theoretical part describes technologies, which have been used for development. The practical part is summarized all the functional and non-functional requirements and on this basis describes functional specification. The conclusion is focused on the implementation of the two main parts of the application, which describes the main features.

Keywords: Web application, online insurance, PHP, Laravel

Rád by som poďakoval svojmu vedúcemu diplomovej práce Ing. Petrovi Šilhavému, Ph.D. za odborné vedenie, podnetné rady, informácie a trpezlivosť, ktoré mi poskytoval počas spracovávania mojej diplomovej práce. Ďalej chcem poďakovať celému projektovému tímu, za úsilie a spoluprácu na projekte a jedno špeciálne poďakovanie patri mojej priateľke a rodine, ktorí mi boli počas štúdia vždy veľkou oporou.

Motto:

“Váš čas je obmedzený, tak ho nemárňte žitím života niekoho iného. Nenechajte sa vlákať do dogmy – teda žiť na základe výsledkov myslenia iných ľudí. Nenechajte v hluku názorov iných, utopiť svoj vlastný vnútorný hlas. A to najdôležitejšie, majte odvahu nasledovať svoje srdce a intuíciu. Všetko ostatné je druhoradé.“

Steve Jobs (1955 - 2011)

OBSAH

ÚVOD	10	
I	TEORETICKÁ ČÁST	11
1	TECHNOLÓGIE POUŽITÉ PRE IMPLEMENTÁCIU	12
1.1	PHP.....	12
1.2	LARAVEL.....	12
1.3	ANGULARJS	13
1.4	CASCADING STYLE SHEETS (CSS)	13
1.5	MYSQL	14
1.6	SIMPLE OBJECT ACCESS PROTOCOL (SOAP)	14
1.7	GIT MANAGMENT	14
II	PRAKTICKÁ ČÁST	16
2	ANALÝZA POŽIADAVKOV	17
2.1	FUNKČNÉ POŽIADAVKY	17
2.1.1	Návštevník.....	17
2.1.2	Aplikácia	17
2.1.3	Administrátorské prostredie (PCT).....	18
2.2	NEFUNKČNÉ POŽIADAVKY.....	19
2.2.1	Záloha.....	19
2.2.2	Doby odozvy systému	19
2.2.3	Rýchlosť prenosu dát a odozvy	20
2.2.4	Výkon	20
2.2.5	Zotavenie po havárii.....	20
2.2.6	Podporované prehliadače	20
2.2.7	Vývojové nástroje	21
3	FUNKČNÁ ŠPECIFIKÁCIA	22
3.1	POPIS APLIKÁCIE	22
3.2	ARCHITEKTÚRA.....	22
3.2.1	Logicko-aplikačná architektúra.....	23
3.2.2	Fyzicko-aplikačná architektúra	25
3.2.3	Architektúra infraštruktúry.....	26
3.3	INTEGRÁCIA	27
3.3.1	Komunikačné rozhranie	27
3.3.1.1	SOAP Rozhranie.....	27
3.4	TESTOVACIE STRATÉGIE.....	31
3.4.1	Funkčné testovanie.....	31
3.4.2	Test nasadenia	31
3.4.3	Výkonnostné a penetračné testy.....	31
3.4.4	Failover testovanie	32
3.5	PRÍPADY UŽITIA.....	32
3.5.1	Aktéri.....	32
3.5.2	Prípady použitia	33
3.5.2.1	UC01 Finálny návrh poistenia	33
3.5.2.2	UC02 Vytvorenie novej poistky	34

3.5.2.3	UC03 Otvorenie existujúcej poistky.....	35
3.5.2.4	UC04 Doladenie poistky.....	35
3.5.2.5	UC05 Doladenie produktov poistky	35
3.5.2.6	UC06 Kontaktujte ma	36
3.5.2.7	UC07 Vyhľadať agenta.....	37
3.5.2.8	UC012 Dashboard manager.....	38
3.5.2.9	UC013 Media manager.....	38
3.5.2.10	UC014 Configuration manager.....	39
3.5.2.11	UC015 Products manager	40
3.5.2.12	UC016 Parameters manager.....	41
3.5.2.13	UC017 Content manager.....	41
3.5.2.14	UC018 Backup manager	42
4	IMPLEMENTÁCIA.....	44
4.1	PCT.....	44
4.1.1	Configuration	47
4.1.1.1	Web Services	47
4.1.1.2	Correlations.....	49
4.1.1.3	FastQuotations	50
4.1.2	Products.....	51
4.1.2.1	Product Categories	51
4.1.2.2	Products	52
4.1.3	Parameters	53
4.1.3.1	Product Parameters	54
4.1.3.2	Visitor a Contract Parameters	55
4.1.4	Page Content	56
4.1.4.1	Details	57
4.1.4.2	First Page	58
4.1.4.3	FastQuotation Page	58
4.1.4.4	Quotation Page.....	58
4.1.4.5	Summary Page	58
4.1.4.6	Finish Page.....	59
4.2	EXAMBLER.....	59
4.2.1	HTTP smerovanie	59
4.2.2	Brána	60
4.2.3	Backend.....	60
4.2.4	Repozitáre	61
4.2.5	Služby.....	62
4.2.5.1	Pomocné funkcie.....	62
4.2.5.2	Placeholders systém.....	63
4.2.5.3	Update Systém	64
4.2.5.4	Mail systém.....	64
4.3	FRONTEND.....	66
4.3.1	Úvodná stránka.....	66
4.3.2	Stránka vypočítaných balíčkov	66
4.3.3	Stránka ponuky.....	67
4.3.4	Konfigurácia produktu	68
4.3.5	Sumarizácia ponuky	68
4.3.6	Dokončenie ponuky	69
	ZÁVĚR	71

SEZNAM POUŽITÉ LITERATURY.....	72
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	73
SEZNAM OBRÁZKŮ	74
SEZNAM TABULEK.....	76

ÚVOD

Internet a online služby presakujú do všetkých odvetví a skôr či neskôr sa tomu budú musieť firmy prispôbiť. Nie je tomu inak ako vo svete poistenia. Výber a možnosti poistenia sú v dnešnej dobe obrovské a preto je nutné byť v niečom iný. Vo väčšine prípadov je pre uzatvorenie poisťky nutné, aby klient navštívil poisťovňu osobne. Tam mu agent ponúkne zo spektra balíčkov a produktov, ktoré spolu podľa požiadaviek nakonfigurujú. Táto operácia však môže trvať dlhý čas a zároveň môže vyžadovať viacero návštev poisťovne. Preto ak sa chce znížiť potrebný čas strávený na uzatvorenie poistenia a zvýšiť spokojnosť zákazníkov je možné využiť výhody online služby pre uzatváranie poistenia, kde si zákazník môže v klude premyslieť a nastaviť celú poisťku podľa seba.

Cieľom práce bolo navrhnutie webovej aplikácie, ktorá atraktívnym spôsobom uľahčí ľuďom uzatváranie poistenia cez internet. Aplikácia je navrhnutá tak aby bola ľahko aplikovateľná pre viacero krajín, kde sa používajú rozličné služby poistenia. Pozostáva z troch hlavných častí. Frontendová časť, ktorá musí svojim atraktívnym vzhľadom a jednoduchým ovládaním zaujať čo najväčšie množstvo zákazníkov. Druhú časť tvorí Exambler, ktorý predstavuje logickú časť celej aplikácie a poslednú časť tvorí konfiguračný nástroj (PCT), ktorý ponúka správcovi šikovný nástroj na nastavenie celého prostredia a portfólia.

V teoretickej časti práce som sa zameril na oboznámenie sa s použitými technológiami z ktorých sa aplikácia skladá. V praktickej časti som začal analýzou požiadaviek, ktoré vyplynuli z viacerých mítingov s klientami. Po analýze som popísal funkčnú špecifikáciu celého projektu. A na záver som sa zameril na finálnu implementáciu, kde som priblížil všetky najdôležitejšie súčasti aplikácie a najkritickejšie oblasti vývoja.

I. TEORETICKÁ ČÁST

1 TECHNOLOGIE POUŽITÉ PRE IMPLEMENTÁCIU

1.1 PHP

PHP je programovací jazyk, ktorý pracuje na strane servera. S PHP môžete ukladať a meniť dáta webových stránok. Pôvodný význam skratky PHP bol Personal Home Page. Vznikol v roku 1996, od tej doby prešiel veľkými zmenami. Teraz táto skratka znamená PHP: Hypertext Preprocessor. [1]

Veľká časť jeho syntaxe je prevzatá z C, Javy a Perlu. Cieľom tohto jazyka je umožniť webovým vývojárom rýchlo písať dynamicky generované stránky. [2]

Základná funkcionálna objektovo orientovaného programovania bola pridaná v PHP 3 a vylepšená v PHP 4. Zaobchádzanie s objektmi bolo kompletne prepísané v PHP 5, kde bolo zároveň rozšírené o nové funkcie a vylepšené z hľadiska výkonnosti. V predošlých verziách PHP bolo k objektom pristupované ako k hodnotovým typom. [3]

Všetky vlastnosti, ktoré počas svojej evolúcie poskytoval skriptovací jazyk PHP, z neho vďaka obľúbenosti u vývojárov vytvorili neoddeliteľnú súčasť webových serverov.

Aj vďaka tomu má dnes zastúpenie na drvivej väčšine internetových stránok. Podľa prieskumu spoločnosti W3Techs sa nachádza až na 80 percentách webových stránok, pričom takmer všetky fungujú na piatej generácii jazyka – PHP 5. [4]

1.2 Laravel

Laravel je pomerne nový PHP framework ktorý sa už aj po tak krátkej dobe radí medzi jeden z najpopulárnejších PHP frameworkov.

Laravel je aplikačný web-framework s výraznou, elegantnou syntaxou. Pokúša sa odstrániť nepríjemnosti vývoja a uľahčuje bežné úlohy používané vo väčšine webových projektov, ako je napríklad overovanie, smerovanie, sessions a caching.

Laravel si kladie za cieľ, aby bol proces vývoja potešujúci pre vývojárov, bez obetovania funkčnosti aplikácie. Za týmto účelom sa pokúsili spojiť to najlepšie z toho, čo videli u iných webových frameworkov, vrátane frameworkov realizovaných v iných jazykoch, ako je Ruby on Rails, ASP.NET MVC a Sinatra.

Laravel je jednoduchý ale poskytuje výkonné nástroje potrebné pre veľké, robustné aplikácie. [5]

1.3 AngularJS

AngularJS je javascriptový webový framework, ktorý sa zameriava na tvorbu single-page aplikácií. Tieto aplikácie sú tvorené pomocou HTML kódu, do ktorého sú vložené špeciálne formátovacie značky, ktoré určujú, aké operácie či dáta majú byť na dané miesto vložené. Tento spôsob sa označuje ako data-binding.

AngularJS je voľne dostupný na webe pod MIT licenciou. V súčasnej dobe je najväčším prispievateľom spoločnosť Google, ktorá tento framework v roku 2009 založila.

AngularJS bol navrhnutý tak, aby pomohol tvorcom stránok oddeľovať zobrazovaciu logiku od aplikačnej logiky, bez väčšej znalosti programovania. Pre tento účel AngularJS využíva návrhový vzor Model-View-Controller - prípadne alternatívny vzor Model-View-ViewModel. Vďaka tomuto abstraktnému oddeleniu je väčšina aplikačnej logiky obsiahnutá v modeli či Controlleri. Tento controller alebo model je následne vložený do \$ scope danej view šablóny, ktorá už môže používať definované premenné a pomocou špeciálnych direktív ich vypisovať či s nimi akokoľvek inak manipulovať.

AngularJS taktiež stráži zmeny, ktoré sa dejú na premenných definovaných pre daný \$ scope, čo je dosiahnuté pomocou takzvaného dirty-checking, ktorý kontroluje či sa nová hodnota líši od starej. Ak sa hodnota premennej zmenila, AngularJS overí, či je skutočne potrebné zmeniť DOM a aktualizuje view template do najnovšieho stavu. Táto funkcionálna prispieva k tomu, že nie je potreba sa starať o znovu vykresľovanie a stráženie zmien a stačí sa len zamerať na výsledný vzhľad aplikácie. [6]

1.4 Cascading Style Sheets (CSS)

Kaskádové štýly alebo CSS je všeobecné rozšírenie jazyka HTML. Konzorcium W3C označuje CSS ako jednoduchý mechanizmus na vizuálne formátovanie internetových dokumentov. Štýly umožnili oddeliť štruktúru HTML alebo XHTML od vzhľadu. Prvá verzia CSS (CSS level 1) vznikla už v roku 1996 a umožňovala prácu s písmami, okrajmi a farbami. V roku 1998 bola doplnená o nové možnosti a vznikol CSS level 2. V súčasnosti je podporovaný vo všetkých novších prehliadačoch (Internet Explorer, Mozilla, Opera, Netscape, Safari). Aktuálna verzia je CSS level 3.

Pomocou kaskádových štýlov sa vytvárajú štruktúrované dokumenty, teda oddeľuje sa obsah dokumentu (HTML) od jeho vzhľadu (CSS). Získa sa tým prehľadný a jednoduchý kód. CSS je možné presunúť do externých súborov, zmenší sa tým dátová veľkosť a dá sa jedným súborom zmeniť celý štýl stránky. CSS zaručuje rovnaké vykresľovanie vo všetkých prehliadačoch. Avšak nie vždy je jednoduché dosiahnuť optimalizovaný kód CSS pre celú škálu prehliadačov. [7]

1.5 MySQL

MySQL je viacvláknový, viac užívateľský SQL relačný databázový server. Je podporovaný na viacerých platformách (ako Linux, Windows či Solaris) a je implementovaný vo viacerých programovacích jazykoch ako PHP, C++ či Perl. Databázový systém je relačný typu DBMS (database management system). Každá databáza je v MySQL tvorená z jednej alebo z viacerých tabuliek, ktoré majú riadky a stĺpce. V riadkoch sa rozoznávajú jednotlivé záznamy, stĺpce udávajú dátový typ jednotlivých záznamov, pracuje sa s nimi ako s poľami. Práca s MySQL databázou je vykonávaná pomocou takzvaných dotazov, ktoré vychádzajú z programovacieho jazyka SQL (Structured Query Language). [8]

1.6 Simple Object Access Protocol (SOAP)

SOAP poskytuje jednoduchý, rozšíriteľný a bohatý XML rámec na výmenu správ, určený pre aplikačné protokoly vyšších úrovní. Ponúka zvýšenú interoperabilitu v distribuovaných heterogénnych prostrediach. SOAP je jednoduchý protokol, určený na výmenu štruktúrovaných informácií v decentralizovanom, distribuovanom prostredí.

SOAP používa technológie XML na definovanie rozšíriteľného komunikačného rámca poskytujúceho štruktúru správ, ktoré môžu byť vymieňané prostredníctvom množstva základných protokolov. Rámec bol navrhnutý tak, aby bol nezávislý od každého konkrétneho programovacieho modelu a iných špecifických sémantik jednotlivých implementácií.

1.7 GIT management

Prvá oficiálna verzia GITU bola uvoľnená 7. apríla 2005. Otcem gitu je Linus Torvalds, jeden z hlavných vývojárov Linux kernelu. Git je prispôbený na to, aby dokázal pracovať s veľkými projektmi (jadro Linuxu je toho dôkazom) a aby dokázal zabezpečiť každé-

mu vývojárovi maximálne pohodlie (to znamená rýchle naučenie, nevyžaduje pripojenie k internetu a podporuje nelineárny vývoj).

Čo nám teda vlastne Git prináša? Môžeme jednoducho zákazníkom publikovať náš projekt vo verziách, ktoré nám vyhovujú. Zákazníkmi bude vnímaný iba ako ďalšia verzia, aj keď medzi predchádzajúcou a aktuálnou verziou môže byť napríklad 10 vývojárskych verzií. Dáva nám jednoduchý prístup k celej histórii projektu. Môžeme sa ľubovoľne v histórii presúvať a máme prehľad o tom, kto vytvoril akú časť kódu a čo sa konkrétne zmenilo. Vďaka vetveniu môžeme vytvárať novú funkcionálnosť bez toho, aby to malo akýkoľvek vplyv na zvyšok projektu. Ak zlyhá, nie je nič jednoduchšie, ako tento pokus nezahrnúť do projektu, vrátiť sa na pôvodné miesto v histórii a začať znovu. Projekt zostane po celý čas v rovnakom stave.

Prvou dôležitou vecou v gite je spôsob ukladania súborov. Git každý súbor uloží len raz a potom ukladá len tzv. Snapshoty. V každom COMMIT (pomyselný uzol systému) sú uložené všetky súbory ako snapshot. To dáva GITU výhody oproti ostatným verziovacím systémom (PERFORCE, CVS, Bazaar), ktoré ukladajú súbor vždy pri jeho zmene. Okrem toho Git ukladá všetky súbory binárne. Je teda možné ukladať aj netextové súbory (napríklad obrázky) a Git poskytne rovnaké pohodlie. Veľkosť repozitára pritom zostane prakticky rovnaká. Potom už závisí viac od formátu súboru, či sa po malej zmene zmení celý binárny súbor, alebo len časť.

Do gitu je tiež veľmi silno zakomponovaná integrita. Pre každý súbor alebo priečinok Git spočíta kontrolný súčet, ktorý sa nazýva SHA-1 hash (40 miestne hexadecimálne číslo). Nemôžete teda zmeniť alebo poškodiť súbor, bez toho aby o tom Git nevedel. Git interne neukladá súbory podľa mena, ale práve podľa tohto kontrolného súčtu. [9]

II. PRAKTICKÁ ČÁST

2 ANALÝZA POŽIADAVKOV

Na základe prieskumu a požiadavkov zákazníka boli špecifikované nasledujúce požiadavky.

2.1 Funkčné požiadavky

2.1.1 Návštevník

- Návštevník bude mať možnosť si vybrať z balíčkov, ktoré sú vypočítané zo vstupných parametrov.
- Návštevník bude môcť pridávať a odoberať produkty z kruhu funkciou Drag&Drop.
- Návštevník si bude môcť nakonfigurovať produkt podľa svojich potrieb.
- Ak bude mať návštevník rozpracovanú ponuku. Môže sa k nej kedykoľvek vrátiť po zadaní svojho vygenerovaného ID.
- Návštevník bude mať možnosť použiť 2 druhy ukončenia ponuky:
 - Kontaktujte ma: v tomto prípade návštevník vyplní formulár so svojimi kontaktnými údajmi a odošle do systému, kde bude objednávka spracovaná a zákazník bude následne kontaktovaný.
 - Nájdi agenta: v tomto prípade bude návštevníkovi vygenerovaná mapa podľa miesta, ktoré zadal na vstupe a ponúkne mu zoznam agentov kde môže svoju poistku dokončiť.

2.1.2 Aplikácia

- Aplikácia bude mať tri nastaviteľné vstupné parametre, z ktorých budú počítané ponuky pre zákazníkov.
- Vstupné parametre budú dátum narodenia, miesto bydliska a povolanie.
- Aplikácia bude podporovať maximálne 12 produktov pre jednu ponuku.
- Roztriedenie produktov v ponuke bude možné rozdeliť medzi maximálne 3 kategórie

- Aplikácia bude spojená s jadrom Agent systému takže agent môže otvoriť ponuku zadáním jedinečného kódu.
- Aplikácia bude ponúkať tri druhy parametrov:
 - Visitor Parameters
 - Products parameters
 - Contract parameters
- Pre každý parameter bude možné zvoliť z týchto typov:
 - Slider
 - Text
 - Number
 - Checkbox
 - Multicheckbox
 - Radiobutton
 - Select box
 - Datepicker

2.1.3 Administrátorské prostredie (PCT)

K aplikácii bude vytvorené externé administrátorské prostredie (PCT), ktoré bude umožňovať:

- Vytvoriť si vlastný Dashboard pre prehľad o stave aplikácie.
- Správa obrázkov a dokumentov použitých na stránke.
- Manažment aplikácie, kde bude možné nakonfigurovať všetky FastQuotations, kategórie, produkty, parametre a webový obsah pre jednotlivé stránky.
- Nastavenie emailov a emailových šablón.
- Nastavenie administrátorov.
- Správa zálohovania.

2.2 Nefunkčné požiadavky

2.2.1 Záloha

- Zálohovanie systému sa vykonáva pomocou RDC. Stratégia zálohovania obsahuje dennú súborovú zálohu a databázové prírastkové zálohy plus týždenné zálohy obrazu disku s rotáciou 3 týždne.

2.2.2 Doby odozvy systému

Očakávaný priemerný čas odozvy klientskej aplikácie je < 3s na akciu používateľa. Aby sa dosiahlo tejto očakávanej doby odozvy (pre 90% žiadostí):

Tabuľka 1 Doby odozvy systému

Vonkajšia komunikácia	
System	Čas odozvy
OE backend	< 1500 ms
ExAmbler	< 1000 ms
Vnútorňa komunikácia	
Databáza	< 200 ms
PHP aplikácia	< 700 ms
Sieťová infraštruktúra	< 100 ms
Rendering prehliadača	< 500 ms

Tabuľka 2 Očakávaná priemerná doba odozvy pre hlavné užívateľské interakcie.

Akcia	Predpokladaná doba odozvy (Z pohľadu užívateľa)
Fast quotations Začiatok: návštevník odošle vstupne údaje Koniec: fast quotations sú zobrazené	< 3000 ms
Recalculation Začiatok: zmení produkt v kruhu Koniec: načíta sa nová cena	< 2000 ms
Quotation summary Začiatok: návštevník klikne na “Quotation Summary” button	< 3000 ms

Koniec: Quotation Summary stránka sa vykreslí	
Načítanie uloženej quotation Začiatok návštevník vloží Quotation ID Koniec: "Quotation" stránka sa vykreslí	< 3000 ms

2.2.3 Rýchlosť prenosu dát a odozvy

Rýchlosť prenosu dát podľa vrstiev:

- Internetová linka: < 200Mbit
- LAN požiadavky: < 1gbit; > 5ms
- MPLS požiadavky: < 20Mbit (dedikovaný); > 20ms

2.2.4 Výkon

Produkčné prostredie musí byť schopné obslúžiť 1000 súbežných užívateľov s dobou odozvy, ako je definovaná vyššie.

2.2.5 Zotavenie po havárii

Zotavenie po havárii je poskytované ako služba od RDC. V neskorších fázach projektu, po úspešnom nasadení projektu do produkcie a dostatočnom otestovaní, by mal byť vypracovaný plán obnovy pre:

- Obnovenie infraštruktúry
- Obnovenie nainštalovaných aplikácií
- Obnovenie najnovších dostupných záloh

2.2.6 Podporované prehliadače

Aplikácie by mala byť verejne prístupná a preto požiadavka je podporiť širokú škálu prehliadačov a klientskych zariadení. Na druhú stranu, cieľom je vytvoriť webovú aplikáciu, čerstvo vyzerajúci v modernom duchu. Nasledujúce verzie prehliadačov majú byť podporované:

- Internet Explorer v9 a vyššie
- Chrome - posledné 4 stabilné verzie
- Firefox - posledné 4 stabilné verzie

- Safari - posledné 2 stabilné verzie

Mobilné zariadenia a ich operačné systémy:

- Android 4.x – 5.x
- iOS 7.1, 8.4, 9.2

Vzhľadom k nekontrolovateľnej širokej škále prístrojov a rôznych OS s úpravami od výrobcu sa vývojový tím zaväzuje za podporu nasledujúcich zariadení:

- iPhone 5, 6, 6s
- iPad 2 and above,
- Samsung Galaxy S3-6
- Google Nexus 5

2.2.7 Vývojové nástroje

Nástroje, ktoré majú byť použité vývojovým tímom:

- | | |
|------------------------|---------------------------------------|
| • Maven v3.0.5 | - build, dependency management (Java) |
| • Jenkins | - build, deployment management |
| • Node / NPM | - dependency management (JavaScript) |
| • Grunt JS v0.1.13 | - build management (JavaScript) |
| • Git | - source code management |
| • Liquibase | - database source control |
| • Sonar | - code quality assurance |
| • Atlassian JIRA | - ticketing |
| • Atlassian Confluence | - WIKI, zdieľanie znalostí |
| • TestNG | - unit testing |
| • Mockito | - mocking backend services |
| • ngdoc | - documenting AngularJS development |
| • Sublime | - frontend development IDE |
| • IntelliJ | - Java development IDE |

3 FUNKČNÁ ŠPECIFIKÁCIA

Táto kapitola obsahuje funkčné špecifikácie pre softvér aplikácie. Účelom tohto dokumentu je poskytnúť prehľad hlavných funkcií systému pre zákazníkov v regióne CEE.

Zákazníci by sa mali odkazovať na tento dokument pri špecifikácii požiadaviek a zmien pre nasadenie aplikácie.

3.1 Popis aplikácie

Hlavné využitie aplikácie pre zákazníka (poist'ovňu) je zjednodušenie procesu uzatvárania online poistenia. Systém je navrhnutý ako on-line aplikácia prístupná prostredníctvom webového prehliadača. Aplikácia poskytuje jednoduchú a prehľadnú formu produktov, ktoré si návštevník môže prispôbovať podľa svojich potrieb. Všetky zmeny sa automaticky ukladajú, takže je možnosť sa kedykoľvek vrátiť a upraviť svoje poistenie. Celý tento online systém poistenia má za úlohu zvýšiť počet klientov a svojím atraktívnym a inovatívnym prístupom ponúknuť vyšší užívateľský komfort. Aplikácia je zároveň prepojená s portálom Agenta, ktorý má prehľad a prístup k vytvoreným poistkám.

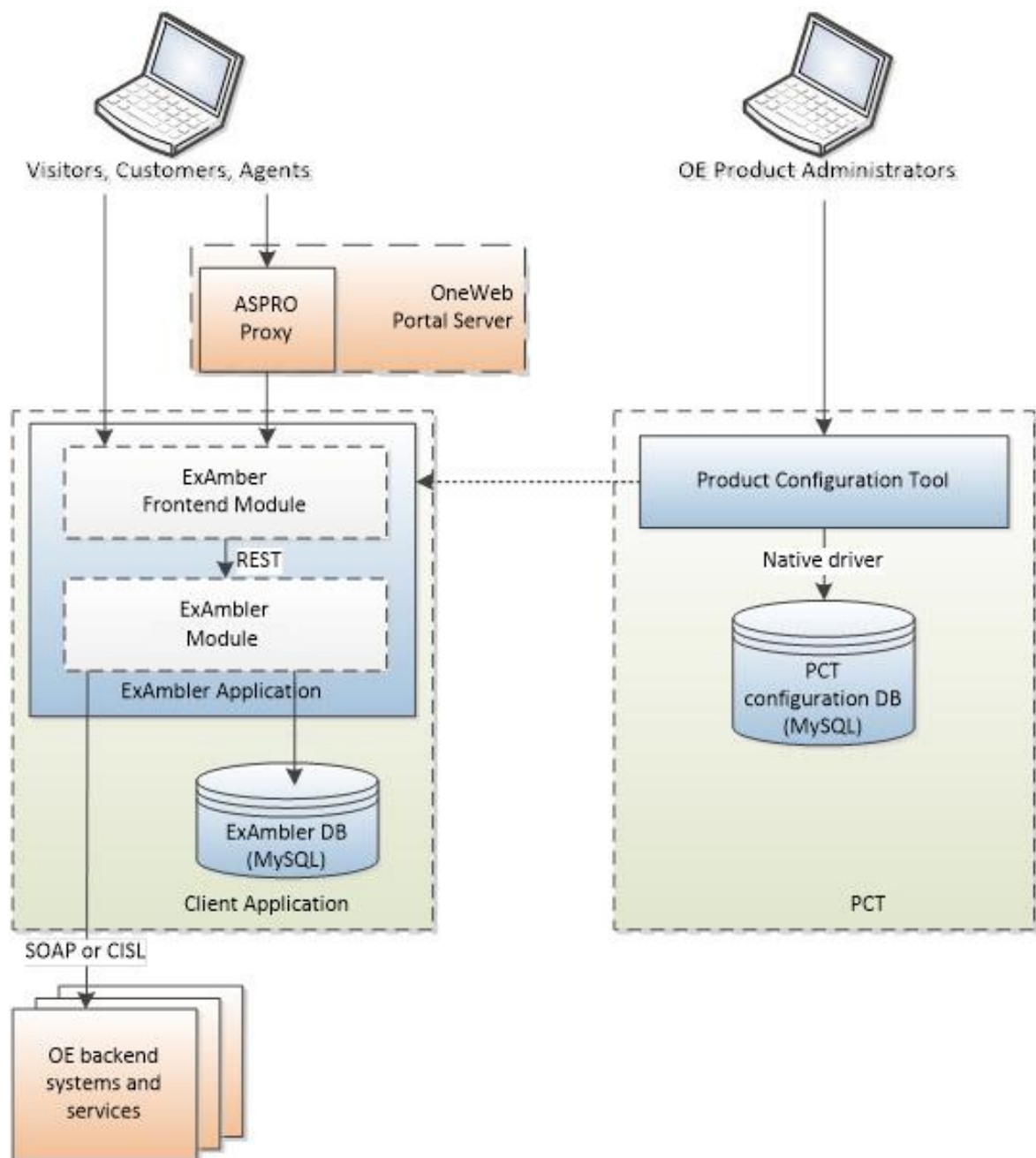
Celý systém pokrýva nasledujúce oblasti:

- Vytvorenie a konfigurovanie návrhu poistenia
- Re-konfigurovanie existujúceho návrhu poistenia
- Online podpísanie poistenia
- Bezpečné platobné metódy spojené s bankovým systémom

3.2 Architektúra

Táto kapitola popisuje architektúru aplikácie z rôznych hľadísk. Od logicko-aplikačnej architektúry, cez fyzickú infraštruktúru až po definovanie rolí a špecifiká jednotlivých komponentov.

3.2.1 Logicko-aplikačná architektúra



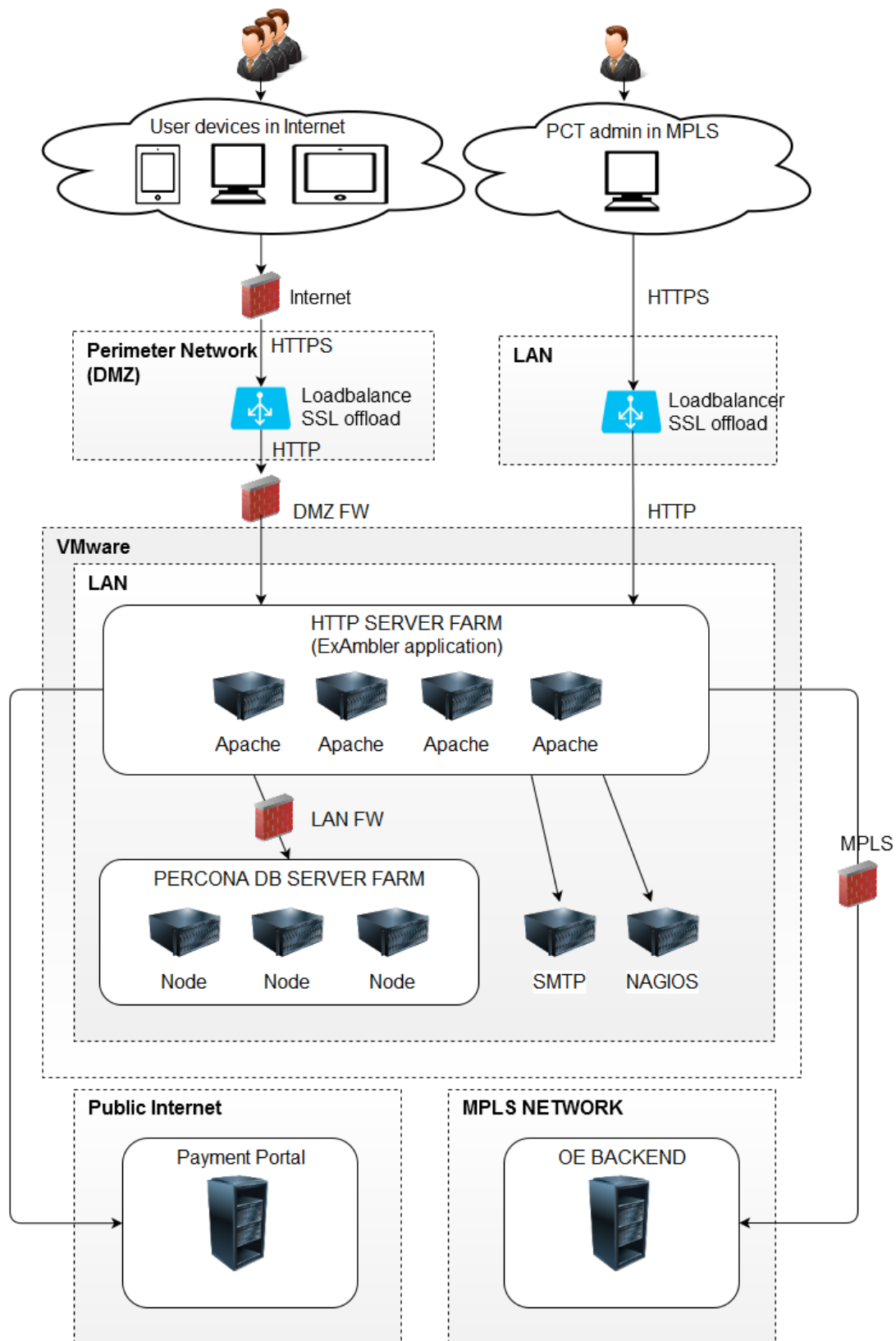
Obrázok 1 Logické súčasti aplikácie a ich interakcie

Tabuľka 3 Popis-logicko aplikačnej architektúri

Komponenta	Popis
ExAmbler Application	PHP aplikácia. Poskytuje používateľské rozhranie a aplikačnú

	<p>logiku klientskej časti AZ1. Skladá sa z 2 hlavných častí:</p> <ul style="list-style-type: none"> • Frontend Module • Java Module
ExAmbler Frontend Module	<p>Poskytuje používateľské rozhranie aplikácie, postavený na AngularJS a niekoľko ďalších knižníc s komponentmi.</p> <p>Aplikácia by mala byť prístupná pre návštevníkov buď :</p> <ul style="list-style-type: none"> • Priamo ako samostatná webová aplikácia (bez oneWeb) - preferovaný spôsob • Umiestnená do oneWeb portálu, cez konfiguráciu servera ASPRO proxy
ExAmbler Module	<p>Poskytuje aplikačnú logiku a dátovú vrstvu. Komunikuje prostredníctvom služby REST medzi frontendom aplikácie a volá OE backend pomocou SOAP alebo Cisl rozhrania.</p>
ExAmbler DB	<p>Databáza aplikácie. Zodpovedný za uloženie Quotations.</p>
Product Configuration Tool (PCT)	<p>Poskytujúce konfiguráciu celého aplikačného rozhrania pre administrátorov OE. Umožňuje editáciu textov, multimediálnych súborov, popisov produktov, funkcií a parametrov uložených v PCT DB.</p>
PCT DB	<p>Databáza na ukladanie konfigurácií aplikácie.</p>

3.2.2 Fyzicko-aplikačná architektúra

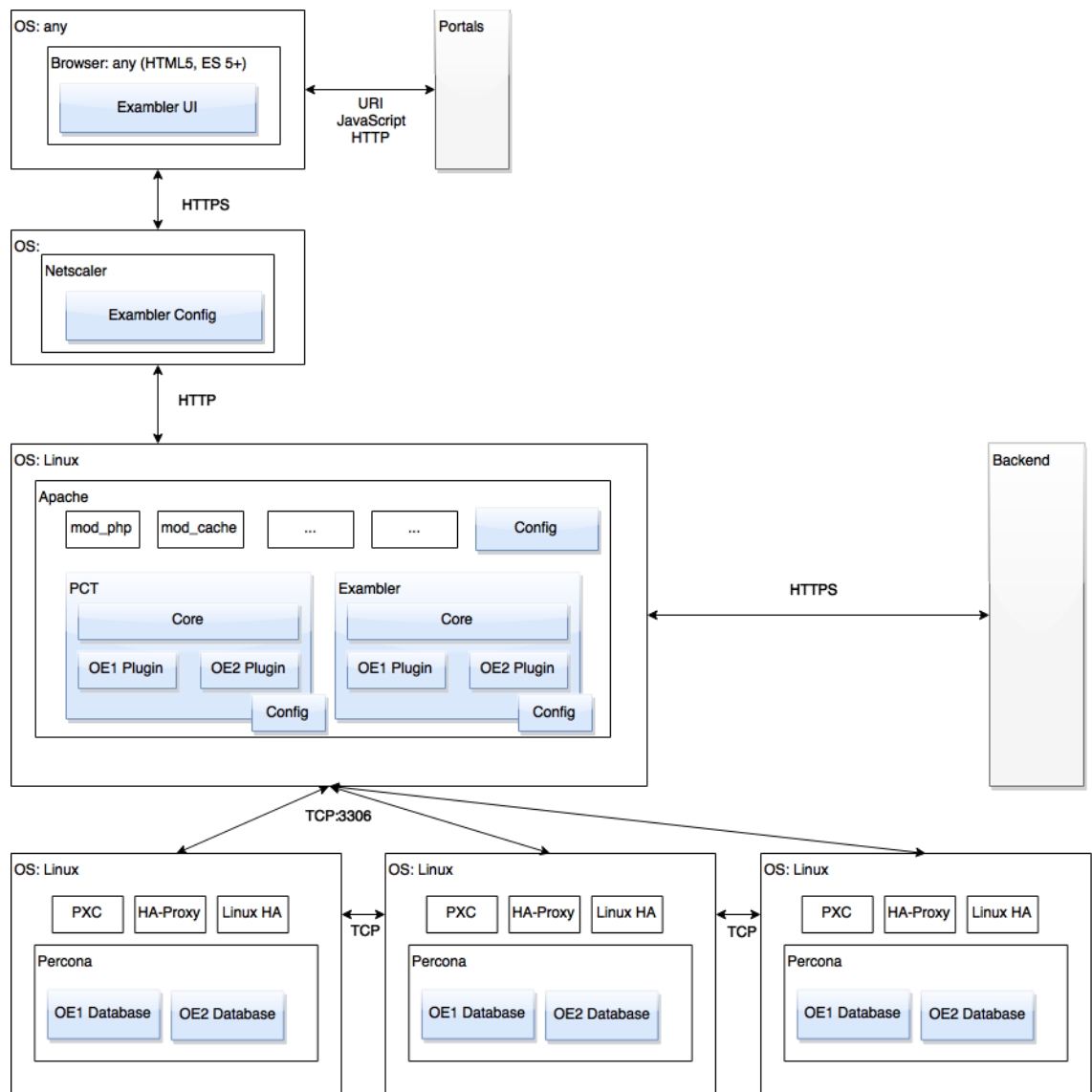


Obrázok 2 Fyzicko-aplikačná architektúra.

3.2.3 Architektúra infraštruktúry

Nasledujúce oddiely opisujú prevádzkovú infraštruktúru prostredia. Definuje podrobné a konkrétne HW prvky. Cieľom je definovať potrebné zdroje (napr.: CPU, pamäť) na úrovni virtuálneho hostiteľa a definovať topológiu nasadenia pre každé prostredie:

- DEV
- UAT
- Hotfix
- PrePROD
- PROD



Obrázok 3 Architektúra infraštruktúry

NetScaler ADC

NetScaler ADC by mal byť nasadený v aktívno-pasívnej konfigurácii umožňujúci vysokú dostupnosť aplikácie tým, že zabezpečí, že vždy bude jedna IP adresa k dispozícii pokrývajúca služby pre klientov, aj keď druhá zlyhá. V takejto situácii, pasívny NetScaler by mal prevziať spoločnú IP a pokračovať vo vyrovnávaní zaťaženia od prichádzajúcich požiadaviek.

Presné charakteristiky systémových zdrojov (napr. spotreba pamäte, využitie procesora) nemožno predvídať vzhľadom k očakávanej zmene a veľa premenných, ale po približných odhadoch na základe skúseností a súčasných charakteristík danej aplikácie, môže tvoriť prvotný základ pre prácu:

- Jedna inštancia webového servera Apache
- Fyzické pridelenie CPU pre hostiteľa: 6 jadier
- Pridelenie pamäte pre hostiteľa: 8 GB
- Disková alokácia pre hostiteľa: 40 GB

3.3 Integrácia

Nasledujúca časť opisuje konkrétne integračné techniky rozhrania.

3.3.1 Komunikačné rozhranie

Aplikácia ponúka 2 možnosti pre komunikáciu medzi OE backendom. K dispozícii je SOAP API s preddefinovaným WSDL alebo CISE rozhranie.

3.3.1.1 SOAP Rozhranie

GetFastQuotation:

Je prvá metóda, ktorá je volaná po tom ako zákazník zadá tri vstupné parametre. Na základe týchto parametrov môže systém pre zákazníka vypočítať ceny balíčkov.

Pre tento výpočet sa na vstupe ešte pripájajú pole všetkých dostupných produktov, ktoré sú nadefinované v PCT a pole preddefinovaných balíčkov taktiež z PCT.

Tabuľka 4 Vstupné parametre pre volanie getFastQuotations

Vstupné parametre:	
Atribút	Popis
dateOfBirth :	Dátum narodenia

professionId :		Profesia
locationId :		Miesto bydliska
products :		
	id :	List všetkých dostupných produktov zadenovaných v PCT
fastQuotations :		
	id :	fastQuotation ID
	products :	List požadovaných produktov ktoré má fastQuotation obsahovať.

Tabuľka 5 Výstupné parametre pre volanie getFastQuotations

Výstupné parametre		
Atribút		Popis
remoteQuotationId :		OE Quotation Id
fastQuotations :		
	id :	fastQuotation id (prebrané zo vstupu)
	price :	cena

GetProductNewUser:

Po tom, čo si zákazník vyberie jeden z balíčkov je zavolaná metóda getProductNewUser.

Táto metóda nám vráti pole všetkých produktov. Každý produkt obsahuje atribúty active, available, editable a price. Na základe týchto atribútov sú potom produkty vykreslené do kruhu.

Tabuľka 6 Vstupné parametre pre volanie getProductNewUser

Vstupné parametre:		
Atribút		Popis
dateOfBirth :		Dátum narodenia
professionId :		Profesia
locationId :		Miesto bydliska
fastQuotationId :		ID vybranej fastQuotation
quotationId :		Interné Quotation ID
remoteQuotationId :		Externé Quotation ID
availableProducts:		
	id :	List všetkých dostupných produktov zadenovaných v PCT
requiredProducts:		
	id :	List požadovaných produktov ktoré musí Quotation obsahovať.

Tabuľka 7 Výstupné parametre pre volanie getProductNewUser

Výstupné parametre		
Atribút		Popis
products:		
	id :	ID produktu
	price :	Cena produktu
	available:	Indikátor dostupnosti produktu
	active:	Indikátor či je produkt aktívny pre danú Quotation
	editable:	Indikátor či môže byť produkt editovateľný
contractParameters:		
	parameterId:	ID parametra
	defaultValue:	Základná hodnota
	arrayType:	Pole hodnôt
		valueId: Key
		valueName: Value
	minMaxType:	
		minValue: Minimálna hodnota parametra
		maxValue: Maximálna hodnota parametra
		step: Veľkosť kroku

GetProductParameters

Posledná metóda, ktorá je potrebná aby bola Quotation kompletne zostavená je volanie getProductParameters. Táto metóda vráti pre požadované produkty ich produktové parametre.

Tabuľka 8 Vstupné parametre pre volanie getProductParameters

Vstupné parametre:		
Atribút		Popis
dateOfBirth :		Dátum narodenia
professionId :		Profesia
locationId :		Miesto bydliska
fastQuotationId :		ID vybranej fastQuotation
quotationId :		Interné Quotation ID
remoteQuotationId :		Externé Quotation ID
products:		
	id :	List požadovaných produktov

Tabuľka 9 Výstupné parametre pre volanie getProductParameters

Výstupné parametre		
Atribút		Popis
products:		

	id :	ID produktu
	parameters:	
	parameterId:	ID parametra
	defaultValue:	Základná hodnota
	arrayType:	Pole hodnôt
	valueId:	Key
	valueName:	Value
	minMaxType:	
	minValue:	Minimálna hodnota parametra
	maxValue:	Maximálna hodnota parametra
	step:	Veľkosť kroku

RecalculateProductPrice

Metóda recalculateProductPrice, slúži na prepočítanie ceny po zmene. Na prepočítanie sa posielajú všetky produkty a parametre bez ohľadu na to či boli zmenené alebo nie. Je to z dôvodu závislostí medzi produktami. Spolu s novou cenou prichádzajú vo výstupe aj tzv. placeholders, ktoré slúžia na vytvorenie dynamického popisu produktu. Rovnaký mechanizmus s placeholderami je použitý aj pre zobrazenie varovnej správy.

Tabuľka 10 Vstupné parametre pre volanie recalculateProductPrice

Vstupné parametre:		
Atribút		Popis
dateOfBirth :		Dátum narodenia
professionId :		Profesia
locationId :		Miesto bydliska
fastQuotationId :		ID vybranej fastQuotation
quotationId :		Interné Quotation ID
remoteQuotationId :		Externé Quotation ID
products:		
	id :	List požadovaných produktov
	active:	Indikátor či je produkt aktívny v kruhu
	updated:	Indikátor či bol od poslednej re-kalkulácie zmenený
	parameters:	
	parameterId:	Parameter ID
	values:	Pole hodnôt
contractParameters:		
	parameterId:	Parameter ID
	values:	Pole hodnôt

Tabuľka 11 Výstupné parametre pre volanie recalculateProductPrice

Výstupné parametre	
Atribút	Popis

products:		
id :		ID produktu
price:		Cena
available:		Indikátor dostupnosti produktu
active:		Indikátor či je produkt aktívny pre danú Quotation
placeholders:		
	search:	Slovo pre vyhľadanie
	replace:	Náhrada vyhľadaného slova
warnMessId		
placeholders:		
	search:	Slovo pre vyhľadanie
	replace:	Náhrada vyhľadaného slova

3.4 Testovacie stratégie

3.4.1 Funkčné testovanie

Cieľom je overiť, či aplikácia má všetky požadované funkcie a spĺňa všetky požiadavky, ktoré boli dohodnuté. Funkčné testovanie sa vykonáva pomocou pripravených testovacích prípadov a manuálneho testovania.

3.4.2 Test nasadenia

Cieľom je overiť, či sú všetky súbory aplikácie schopné nasadenia podľa návodu na inštaláciu.

3.4.3 Výkonnostné a penetračné testy

Cieľom je, aby meranie priepustnosti, výkonnosti a škálovateľnosti aplikácie s databázou spĺňalo požiadavky. Test musí byť vykonaný na základe plánu vývojového tímu a administrátorov v prostredí, ktoré má rovnaké alebo približné parametre ako má produkčné prostredie.

Merania a charakteristiky by sa mali zamerať na:

- Čas odozvy servera vystaveného rastúcemu počtu súbežných užívateľov
- Využitie CPU a pamäte
- Škálovanie (jeden uzol / multi uzol)

3.4.4 Failover testovanie

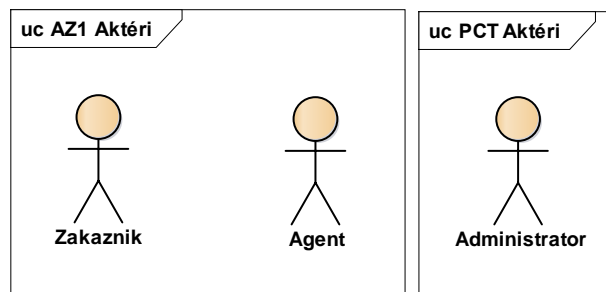
Cieľom je zistiť príčiny zlyhania určitých komponentov. Hlavné testovacie prípady a očakávané výsledky:

- Aktívna zložka NetScaler zlyhá
 - Pasívny NetScaler automaticky preberá.
- Apache Web Server zlyhá
 - NetScaler presmeruje trasy dopravy smerom k pracovnému Apache
- Aktívna inštancia databázy zlyhá
 - Manuálne prepnutie možné s minimálnym prerušením

3.5 Prípady užitia

3.5.1 Aktéri

Podľa zadanej požiadavky boli zvolení títo aktéri:



Obrázok 4 Aktéri aplikácie.

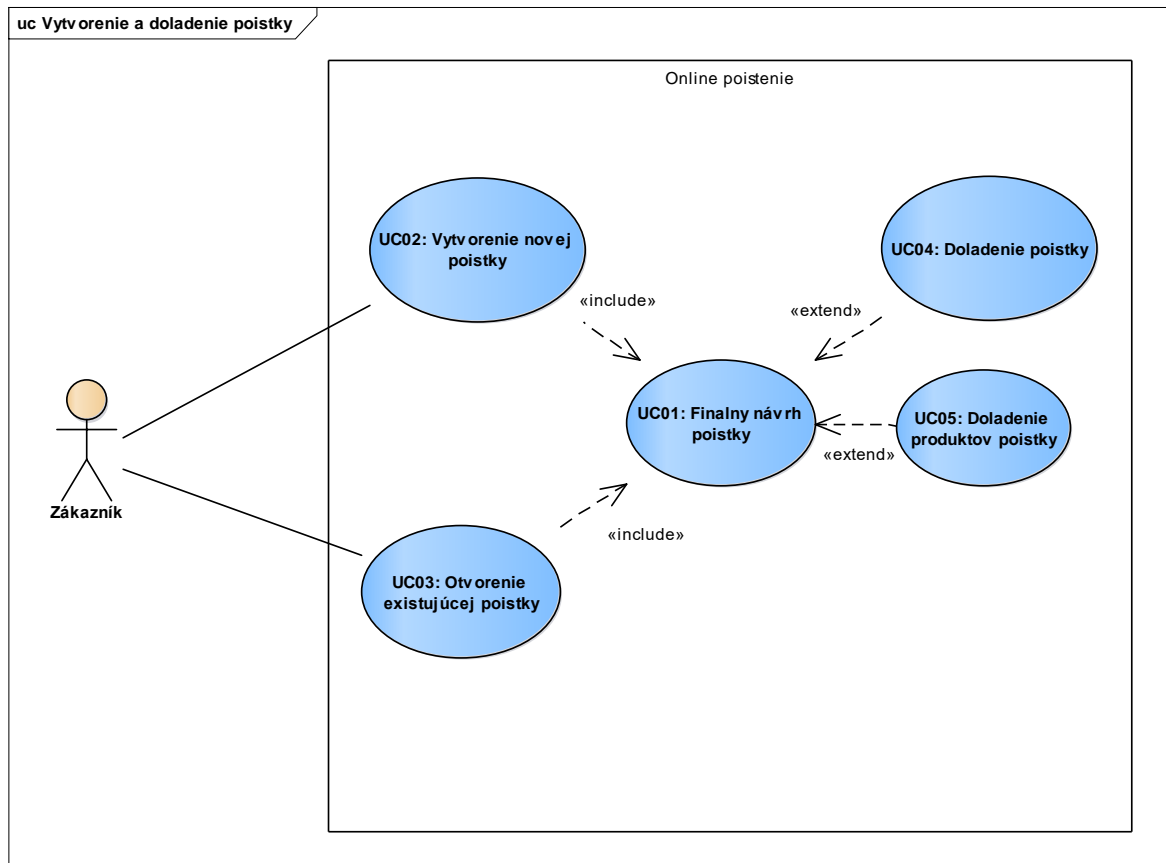
Zákazník – Môže nakonfigurovať svoje nové poistenie, alebo otvoriť existujúce nedokončené. Pre vytvorenie nového musí zadať požadované vstupné parametre.

Agent – Môže upravovať a uzatvárať predkonfigurované poistenia.

Administrátor – Ma prístup k externej časti aplikácie, kde sa nastavujú všetky súčasti systému.

3.5.2 Prípady použitia

Diagram online poistenia v sebe zahrnuje všetky kroky od vytvorenia nového poistenia a nastavenia si ho podľa potrieb až po otvorenie existujúceho poistenia.



Obrázok 5 Diagram prípadu použitia na online poistenie

3.5.2.1 UC01 Finálny návrh poistenia

Názov prípadu použitia	UC01 Finálny návrh poistky
Vstupné podmienky	
Výstupné podmienky	Cenová ponuka bola vytvorená
Aktéri	Zákazník
Scenár	

1. Systém zobrazí ponuku (identifikovateľnú podľa ID) s aktívnymi produktami v kruhu a zvyšok produktov z ktorých možno zvoliť mimo kruhu. Ak je vybraná prázdna ponuka nie sú v kruhu žiadne produkty. V tomto bode môže zákazník:

-
- a. Zákazník môže pridávať a odoberať produkty
 - b. Zákazník môže upravovať produkty
2. Zákazník pokračuje na stránku z prehľadom svojej ponuky
 3. Systém zobrazí prehľad o vybraných produktoch rozdelených podľa kategórií. V tomto bode môže zákazník:
 - a. Zákazník môže odoberať produkty
 - b. Zákazník môže upravovať produkty
 4. Zákazník dokončí prípad použitím zvolením jednej z ukončovacích akcií
 - a. Nájde umiestnenie agentúry
 - b. Odošle požiadavku aby bol kontaktovaný agentom
 - c. Zakúpenie pomocou online podpísania poistenia

Alternatívy	
4.1 Systém informuje zákazníka pred ukončením konfigurácie na vybrané produkty ktoré majú stále základné nezmenené hodnoty	
4.2 Zákazník môže zmeniť základné hodnoty	
4.3 Prípad užitia pokračuje v bode 4	

3.5.2.2 UC02 Vytvorenie novej poistky

Názov prípadu užitia	UC02 Vytvorenie novej poistky
Vstupné podmienky	
Výstupné podmienky	Cenová ponuka bola vytvorená
Aktéri	Zákazník
Scenár	

1. Zákazník zadá vstupné údaje (napr. dátum narodenia, povolanie, mesto)
 2. Systém zobrazí na výber preddefinované cenové balíčky a prázdny balíček
 3. Zákazník vyberie jednu z ponúkaných možností
-

4. Proces pokračuje v UC01 Finálny návrh poistky

3.5.2.3 UC03 Otvorenie existujúcej poistky

Názov prípadu použitia	UC03 Otvorenie existujúcej poistky
Vstupné podmienky	Zákazník má ID existujúcej ponuky
Výstupné podmienky	Cenová ponuka bola upravená
Aktéri	Zákazník
Scenár	
<ol style="list-style-type: none">1. Zákazník vloží ID existujúcej ponuky2. Systém načíta a zobrazí uloženú ponuku3. Proces pokračuje v UC01 Finálny návrh poistky	

3.5.2.4 UC04 Doladenie poistky

Názov prípadu použitia	UC04 Doladenie poistky
Vstupné podmienky	Ponuka je načítaná a zobrazená
Výstupné podmienky	Cenová ponuka bola upravená
Aktéri	Zákazník
Scenár	
<ol style="list-style-type: none">1. Zákazník pridá alebo odstráni produkty z kruhu2. Systém aktualizuje obsah a prepočíta cenu ponuky	

3.5.2.5 UC05 Doladenie produktov poistky

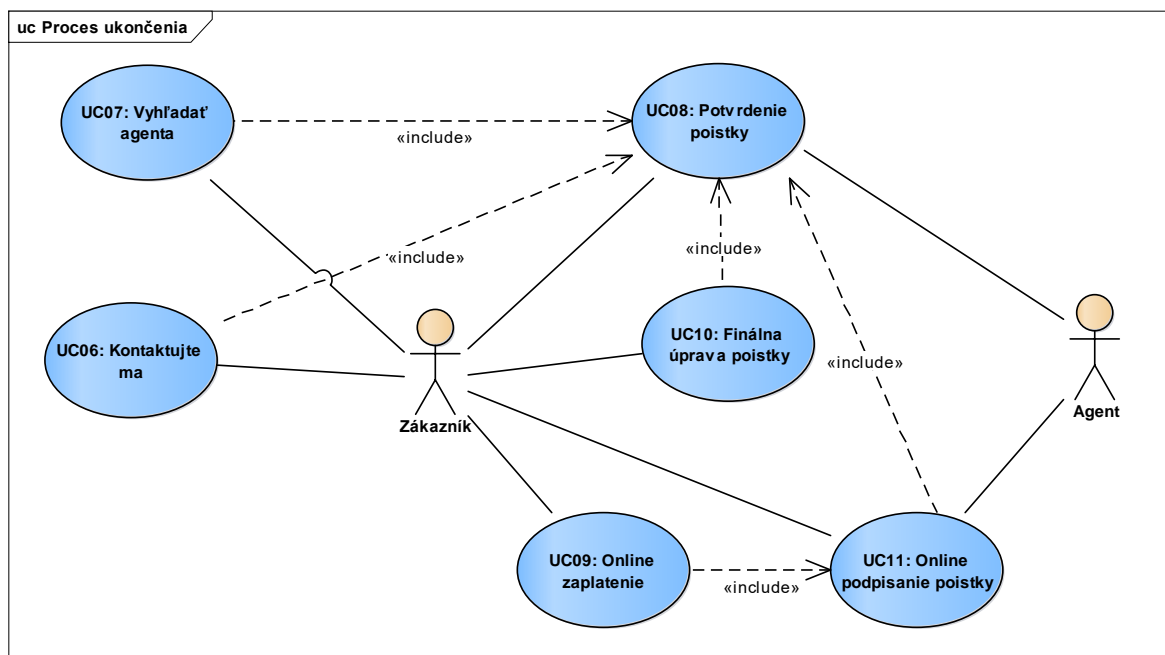
Názov prípadu použitia	UC05 Doladenie produktov poistky
Vstupné podmienky	Ponuka je načítaná a zobrazená
Výstupné podmienky	Produktové parametre boli upravené
Aktéri	Zákazník

Scenár	
<ol style="list-style-type: none"> 1. Zákazník upraví hodnoty produktových parametrov 2. Systém aktualizuje obsah a prepočíta cenu produktu na základe zmenených hodnôt 3. Zákazník potvrdí zmenené parametre 	

Diagram procesu ukončenia v sebe zahrnuje všetky možné metódy ako ukončiť už vytvorené a nastavené poistenie.

Avšak proces ukončenia je stále vo fáze návrhu a analýzy a do tejto chvíle sú špecifikované len dve metódy ukončenia:

- UC06 – Kontaktujte ma
- UC07 – Vyhľadať agenta



Obrázok 6 Diagram prípadu užitia na proces ukončenia ponuky

3.5.2.6 UC06 Kontaktujte ma

Názov prípadu užitia	UC06 Kontaktujte ma
----------------------	---------------------

Vstupné podmienky	Ponuka je načítaná a zobrazená
Výstupné podmienky	Úspešne odoslaný email
Aktéri	Zákazník
Scenár	

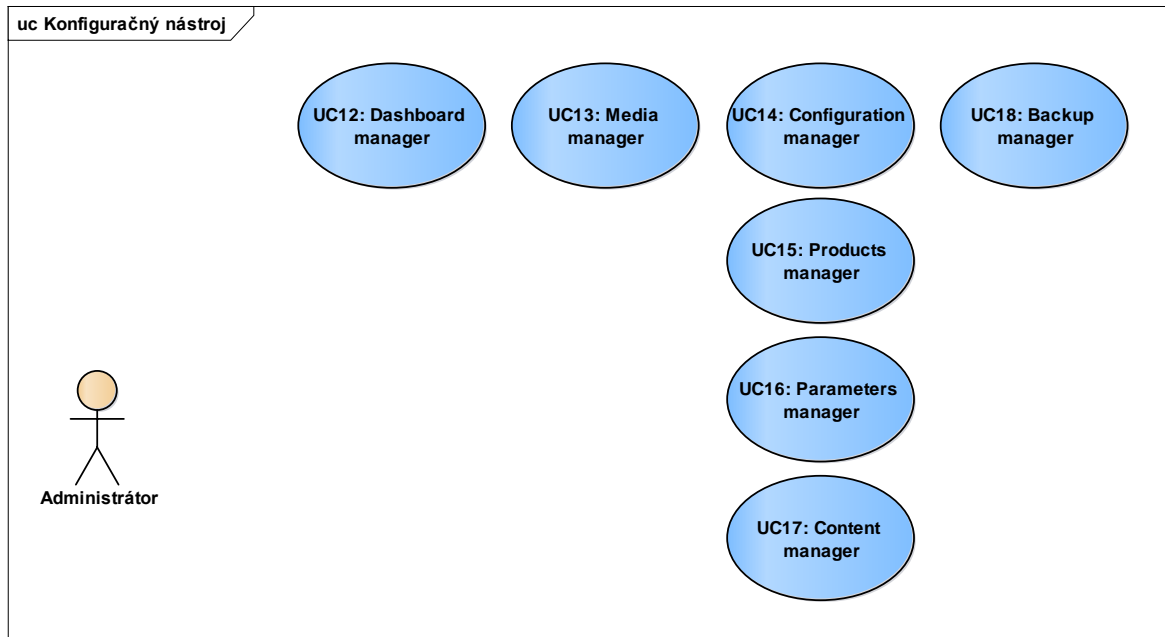
1. Zákazník vyplní všetky kontaktné informácie
2. Systém skontroluje údaje a označí ponuku za ukončenú.
3. Systém odošle informačný email zákazníkovi

3.5.2.7 UC07 Vyhľadať agenta

Názov prípadu použitia	UC07 Vyhľadať agenta
Vstupné podmienky	Ponuka je načítaná a zobrazená
Výstupné podmienky	Úspešne odoslaný email
Aktéri	Zákazník
Scenár	

1. Systém načíta mapu s pobočkami podľa polohy zákazníka
2. Zákazník si vyberie jednu z dostupných pobočiek a zadá svoj email
3. Systém skontroluje údaje a označí ponuku za ukončenú.
4. Systém odošle informačný email zákazníkovi

Diagram pre konfiguračný nástroj PCT v sebe zahrnuje všetky možnosti, ktoré je možné použiť pre konfiguráciu aplikácie.



Obrázok 7 Diagram prípadov užitia pre Administrátorov

3.5.2.8 UC012 Dashboard manager

Názov prípadu užitia	UC12 Dashboard manager
Vstupné podmienky	Prihlásený administrátor
Výstupné podmienky	
Aktéri	Administrátor
Scenár	

1. Administrátor otvorí položku Dashboard
2. Povolené operácie:
 - a. Pridať a konfigurovať widget
 - b. Konfigurovať existujúci widget
 - c. Zmazať widget

3.5.2.9 UC013 Media manager

Názov prípadu užitia	UC13 Media manager
----------------------	--------------------

Vstupné podmienky	Prihlásený administrátor
Výstupné podmienky	
Aktéri	Administrátor
Scenár	

1. Administrátor otvorí položku Media
2. Povolené operácie:
 - a. Nahrávanie nových súborov alebo obrázkov
 - b. Pridať priečinok
 - c. Vyhľadávanie
 - d. Presúvanie
 - e. Vymazávanie
 - f. Filtrovanie podľa (obrázkov, videí, zvukov, dokumentov)
 - g. Zvolenie druhu zobrazenia

3.5.2.10 UC014 Configuration manager

Názov prípadu užitia	UC14 Configuration manager
Vstupné podmienky	Prihlásený administrátor
Výstupné podmienky	Upravené hodnoty boli uložené
Aktéri	Administrátor
Scenár	

1. Administrátor otvorí položku Az1 / Configuration
 - a. / Web Services:
 - Povolené operácie:
 1. Nastavenie typu komunikácie medzi aplikáciou a systémom poisťovne
 2. Zadanie URL cesty systému poisťovne
 3. Nahratie konfiguračného súboru pre mapu agentúr

b. / Correlations

▪ Povolené operácie:

1. Pridať novú koreláciu
2. Upraviť existujúcu koreláciu
3. Zmazať koreláciu ak nie je priradená k žiadnemu parametru

c. / FastQuotations

▪ Povolené operácie:

1. Pridať novú FastQuotation
 2. Upraviť existujúcu FastQuotation
 3. Zmazať FastQuotation
-

3.5.2.11 UC015 Products manager

Názov prípadu použitia	UC15 Products manager
Vstupné podmienky	Prihlásený administrátor
Výstupné podmienky	Upravené hodnoty boli uložené
Aktéri	Administrátor
Scenár	

1. Administrátor otvorí položku Az1 / Products

a. / Products Categories:

▪ Povolené operácie:

1. Pridať novú kategóriu
2. Upraviť existujúcu kategóriu
3. Zmazať kategóriu

b. / Products

▪ Povolené operácie:

1. Pridať nový produkt
-

2. Upraviť existujúci produkt

3. Zmazať produkt

3.5.2.12 UC016 Parameters manager

Názov prípadu užitia	UC16 Parameters manager
Vstupné podmienky	Prihlásený administrátor
Výstupné podmienky	Upravené hodnoty boli uložené
Aktéri	Administrátor
Scenár	

1. Administrátor otvorí položku Az1 / Parameters

a. / Product Parameters:

b. / Contract Parameters

c. / Visitor Parameters

▪ Povolené operácie:

1. Pridať nový parameter

2. Upraviť existujúci parameter

3. Zmazať parameter

3.5.2.13 UC017 Content manager

Názov prípadu užitia	UC17 Content manager
Vstupné podmienky	Prihlásený administrátor
Výstupné podmienky	Upravené hodnoty boli uložené
Aktéri	Administrátor
Scenár	

1. Administrátor otvorí položku Az1 / Page Content

a. / Details:

▪ Povolené operácie:

1. Upraviť Titulok stránky, Meta popis stránky, Meta kľúčové slová, odkaz na cookie
2. Pridať vlastný JavaScript
3. Pridať vlastné CSS
4. Upraviť preklady stránky

b. / First Page

▪ Povolené operácie:

1. Upraviť Slogan, text pre Header, text pre Footer
2. Vybrať obrázok pre pozadie
3. Pridať Video Slideshow
4. Pridať články

c. / Fast Quotation Page

▪ Povolené operácie:

1. Upraviť text pre Header, text pre Body, text pre Footer

d. / Quotation Page

▪ Povolené operácie:

1. Upraviť text pre Header, text pre Body, text pre Footer
2. Vybrať obrázok pre Help obrazovku

e. / Summary Page

▪ Povolené operácie:

1. Upraviť text pre Header, text pre Body, text pre Footer
2. Upraviť texty pre Info Box

3.5.2.14 UC018 Backup manager

Názov prípadu užitia

UC18 Backup manager

Vstupné podmienky	Prihlásený administrátor
Výstupné podmienky	Zip súbor na stiahnutie
Aktéri	Administrátor
Scenár	

1. Administrátor otvorí položku Az1 / Backup Manager

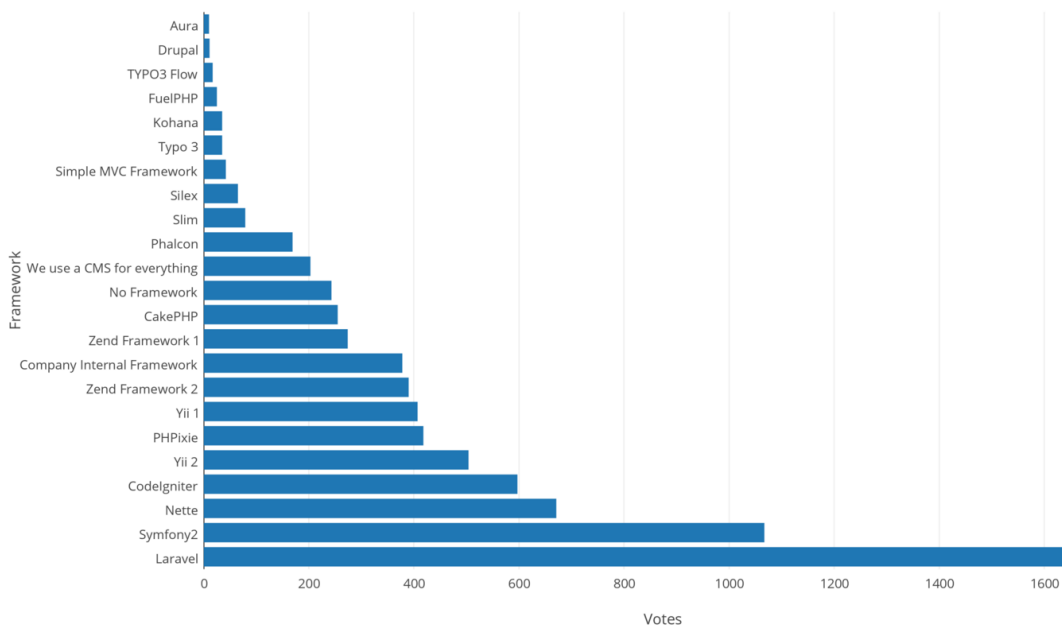
a. / Export

▪ Povolené operácie:

1. Export logov aplikácie
 2. Export konfigurácií
 3. Export nastavení
 4. Plný export
 5. Export pre oneWeb aplikáciu
-

4 IMPLEMENTÁCIA

Ako programovací jazyk, v ktorom má byť aplikácia realizovaná bol vybraný PHP jazyk. Na základe toho bolo prvým krokom vybrať vyhovujúci framework na ktorom sa aplikácia postaví. Po prieskume internetu a zhodnotení sme sa nakoniec zhodli a vybrali framework Laravel, ktorý zaznamenal v posledných rokoch veľký nárast v obľúbenosti.



Obrázok 8 Prieskum najobľúbenejších PHP Frameworkov v roku 2015.

Druhým krokom bolo zostaviť developerské prostredie. V prvej fáze vývoja sme mali len jeden linuxový testovací server na ktorom bola nahratá MySQL databáza, Apache server a keďže aplikácia je rozdelená na viac častí, vytvorili sme GIT repozitáre. Samotný vývoj však prebiehal na lokálnej úrovni jednotlivými programátormi, kde mali spustený svoj vlastný Apache server.

Aplikácia je rozdelená na tri časti. Prvá samostatná časť je PCT a ďalšie dve časti sú Frontend a Exambler.

4.1 PCT

Pre nastavenie aplikácie bolo potrebné mať nejaký konfiguračný CMS systém, ktorý budú môcť potenciálni administrátori jednoducho ovládať. Pre tieto potreby bolo dohodnuté, že nie je nevyhnutné vyvíjať kompletne nový systém a preto sme našli vhodný open source

CMS systém - October CMS, ktorý spĺňal všetky naše požiadavky. Je taktiež celý postavený na frameworku Laravel, má v sebe plugin systém a správu užívateľov.

Po jeho nainštalovaní sme mali priamo splnené 4 funkčné požiadavky:

- Vytváranie si vlastných Dashboardov pre prehľad o stave aplikácie
- Správa obrázkov a dokumentov použitých na stránke
- Natavenie emailov a emailových šablón
- Nastavenie administrátorov

Pre požiadavku na manažment aplikácie už však bolo potrebné vytvoriť vlastný plugin. October CMS však ponúka veľmi detailnú dokumentáciu, kde sú všetky časti prehľadne popísané.

Plugins sú umiestnené v adresári **/plugins**, ktorý je podadresárom aplikácie.

```
plugins/  
  acme/          <=== Author name  
    blog/        <=== Plugin name  
      classes/  
      components/  
      controllers/  
      models/  
      updates/  
      ...  
  Plugin.php     <=== Plugin registration file
```

Obrázok 9 Adresárová štruktúra pluginov.

October obsahuje niekoľko konzolových príkazov, ktoré umožňujú inštalovať, aktualizovať a celkovo urýchliť proces vývoja. Príkazová konzola je súčasťou Laravel frameworku a je nazývaná Artisan.

Pre vygenerovanie nového pluginu a jeho štruktúry sme použili príkaz:

```
php artisan create:plugin AuthorName.PluginName
```

Dôležitá časť pre každý plugin je jeho registračný súbor nazývaný **Plugin.php**. Je to inicializačný skript, ktorý deklaruje základné funkcie a informácie o plugine. Pre našu aplikáciu som nadefinoval tieto hlavné časti:

- Detail pluginu ako sú: meno, ikonka, popis, ktoré sa prejavia v administrátorskej časti menu.
- Navigačné menu. Kde pre každú položku je zadefinované meno, popis, ikonka, kategória, oprávnenie a odkaz.

- Taktiež je možné si vytvoriť vlastné skupiny oprávnení, ktoré sa následne dajú aplikovať na registrovaných administrátorov.
- A poslednú časť predstavuje registráciu pre vlastné widgety pre Dashboard.

```

10 class Plugin extends PluginBase
11 {
12
13     public function pluginDetails()
14     {
15         return [ ... ];
20     };
21     }
22
23     public function registerNavigation()
24     {
25         return [ ... ];
46     };
47     }
48
49     public function registerPermissions()
50     {
51         return [ ... ];
68     };
69     };
70     }
71
72     public function registerReportWidgets(){
73         return [ ... ];
78     };
79     }
80
81 }
82

```

Obrázok 10 Štruktúra registračného súboru Plugin.php.

Po registrovaní nového pluginu prichádzajú na radu časti v ktorých sa bude konfigurovať aplikácia. Keďže October CMS implementuje vzor MVC každá takáto časť obsahuje vlastný model, view a controller.

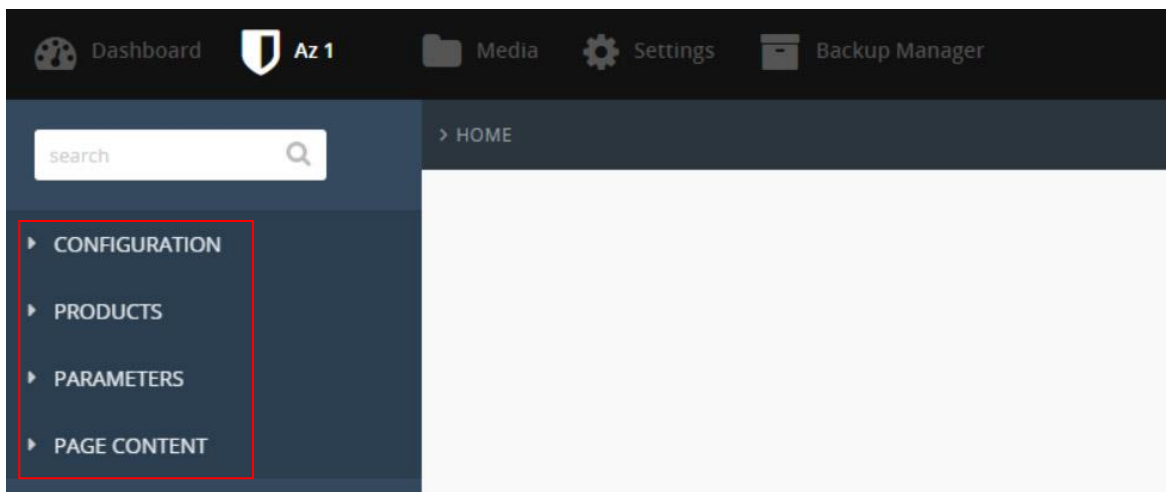
```

plugins/
  acme/
    blog/
      controllers/
        users/
          _partial.htm <=== Controller view directory
          config_form.yaml <=== Controller partial file
          index.htm <=== Controller config file
          Users.php <=== Controller view file
          Plugin.php <=== Controller class

```

Obrázok 11 Základná štruktúra pre controller.

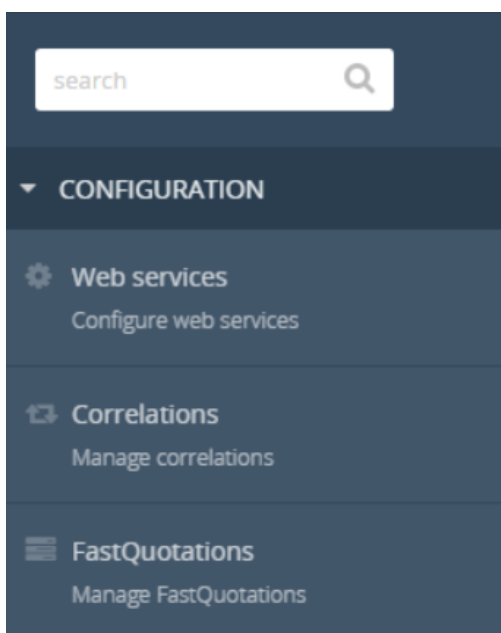
Na vytvorenie nového modelu a controllera som opäť využil CLI Artisan, ktorý za mňa vygeneroval celú základnú štruktúru. Týmto spôsobom som si vytvoril kompletnú štruktúru pre všetky jednotlivé časti, ktoré majú mať svoju funkcionlitu. Z funkčných požiadaviek som si tieto časti rozdelil do logických skupín.



Obrázok 12 Rozdelenie položiek menu do logických skupín.

4.1.1 Configuration

Skupina Configuration obsahuje položky pre globálnejšie súčasti aplikácie ako je napríklad nastavenie typ komunikácie medzi aplikáciami.



Obrázok 13 Menu pre kategóriu Configuration.

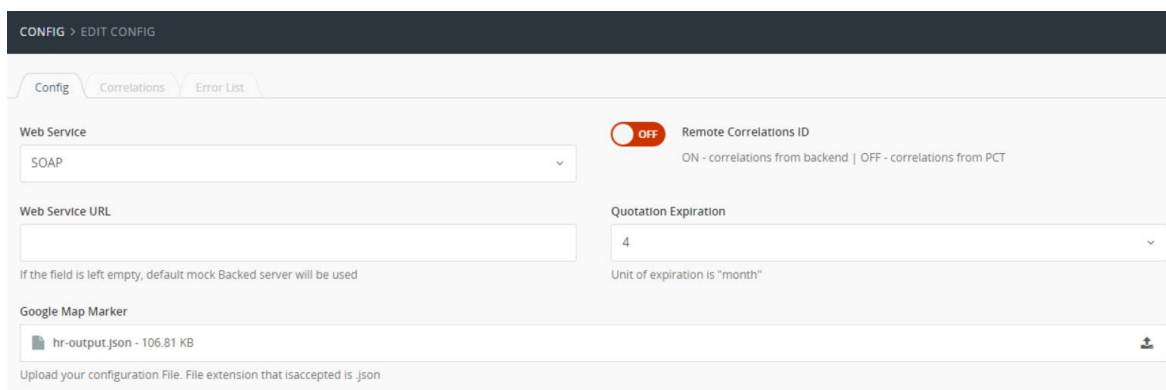
4.1.1.1 Web Services

Jednou z najdôležitejších častí aplikácie je komunikácia. Administrátor si tu môže nastaviť typ komunikácie a adresu produkčného backendu. Okrem toho je tu možnosť nahráť konfiguračný súbor pre Google mapu a nastaviť dĺžku trvania jednej vytvorenej ponuky.

Vytváranie vizuálnej časti backendu je v October CMS veľmi dobre spracované. Slúži na to YAML konfiguračný súbor. V tomto súbore sa nabindujú jednotlivé vstupné polia na model.

```
1 # =====
2 # Form Field Definitions
3 # =====
4 tabs:
5   fields:
6     id:
7       label: ID
8       hidden: true
9       span: left
10      disabled: true
11      tab: Config
12
13     service:
14       label: Web Service
15       span: left
16       attributes: { data-toggle: "tooltip", title: "Select type of Web Service" }
17       type: dropdown
18       tab: Config
19       options:
20         # cisl: Cisl/Rest
21         soap: SOAP
22
23     remote_correlations:
24       label: Remote Correlations ID
25       comment: ON - correlations from backend | OFF - correlations from PCT
26       span: right
27       type: switch
28       tab: Config
29
30     servicepath:
31       label: Web Service URL
32       span: left
33       comment: If the field is left empty, default mock Backed server will be used
34       disabled: false
35       type: text
36       tab: Config
37
38     expire:
39       label: Quotation Expiration
40       span: right
41       attributes: { data-toggle: "tooltip", title: "Value of expiration time is MONTH" }
42       comment: Unit of expiration is "month"
43       type: dropdown
44       tab: Config
45       options:
46         1: 1
47         2: 2
48         3: 3
49         4: 4
50
51     markers:
52       label: Google Map Marker
53       comment: Upload your configuration File. File extension that isaccepted is .json
54       type: fileupload
55       mode: file
56       tab: Config
57       fileTypes: json
```

Obrázok 14 YAML konfigurácia pre sekciu Web Services.



Obrázok 15 Výsledok po nakonfigurovaní.

4.1.1.2 Correlations

Funkcia korelácie slúži na spojenie dvoch parametrov rovnakého typu. Administrátor má možnosť si nadefinovať jednotlivé korelačné skupiny, ktoré v neskoršej fáze vytvárania produktov môže aplikovať.

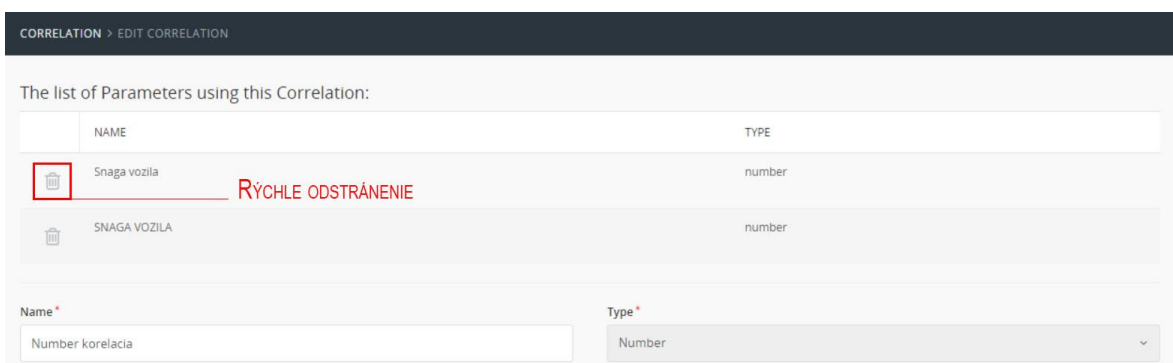
Vytvorenie vizuálnej časti bolo tiež definované pomocou YAML konfiguračného súboru avšak z pohľadu funkčnosti sa pri používaní korelácií môžu naskytnúť neželané situácie ako je napríklad odstránenie korelácie, ktorá je ešte stále priradená k parametrom. Preto som pridal kontrolnú funkciu, ktorá kontroluje či je zvolená korelácia niekde priradená. Ak ju používa nejaký parameter, je zablokované tlačidlo na odstránenie. Pre skvalitnenie použiteľnosti ponúka dané rozhranie priamo odstrániť takéto spojenie.



```
30 public function getParameters()
31 {
32     if (isset($this->params[0])) {
33         $correlationId = $this->params[0];
34
35         $productParameters = Parameter::Where('correlation_id', $correlationId)->get()->toArray();
36         $this->vars['productParameters'] = $productParameters;
37
38         $contractParameters = ContractParameter::Where('correlation_id', $correlationId)->get()->toArray();
39         $this->vars['contractParameters'] = $contractParameters;
40     }
41 }
42 }
```

Obrázok 16 Funkcia načítania všetkých parametrov, ktoré používajú vytvorenú koreláciu.

```
44 public function onClearCorrelationFromProductParameter()
45 {
46     $productParameterId = post('productParameterId');
47
48     Parameter::where('id', $productParameterId)->update(['correlation_id' => 0]);
49 }
50 }
```

Obrázok 17 Funkcia pre rýchle odstránenie korelácie z parametra.



	NAME	TYPE
	Snaga vozila	number
	SNAGA VOZILA	number

Name*

Type*

Obrázok 18 Administrácia korelácie, kde je vidieť zoznam všetkých parametrov ktoré majú v sebe priradenú Number koreláciu.

4.1.1.3 FastQuotations

Predstavuje správu balíčkov, kde administrátor okrem mena a popisu nastavuje hlavne tzv. Cover Balance. Funguje to tak, že si môže vybrať zo zoznamu dostupných produktových kategórii, ktoré sú spracovateľné v inej sekcii administrácie. K tejto kategórii následne priradí informačnú hodnotu 1-5, ktorá určuje výšku pokrytia. Je to v podstate len informačná hodnota pre zákazníka. A ďalšia veľmi dôležitá časť ktorú nastavuje sú produkty, ktoré má daný balíček obsahovať.

Pri programovaní poľa pre výber produktov a nastavení Cover Balance som musel výsledok uložiť do databázy ako JSON objekt. K tomuto účelu existuje funkcia :

```
22 protected $jsonable = ['cover_balance', 'required_products'];
```

Obrázok 19 Hodnoty sú kódované ako JSON pred uložením a následne prevedené na pole po načítaní.

Tento formát mi taktiež v neskoršej fáze vývoja uľahčil prácu pri formátovaní vstupu pre komunikáciu s backendom.

OE CATEGORY ID	VALUE
Life	1
House	1
Vehicle	2

Products

Select: all, none

- Building Insurance
- Personal estate insurance
- Assistance insurance
- Personal insurance
- Accidental disability insurance
- Accident daily charge insurance
- Life Insurance II
- Life Insurance III
- Life Insurance IV
- Life Insurance V

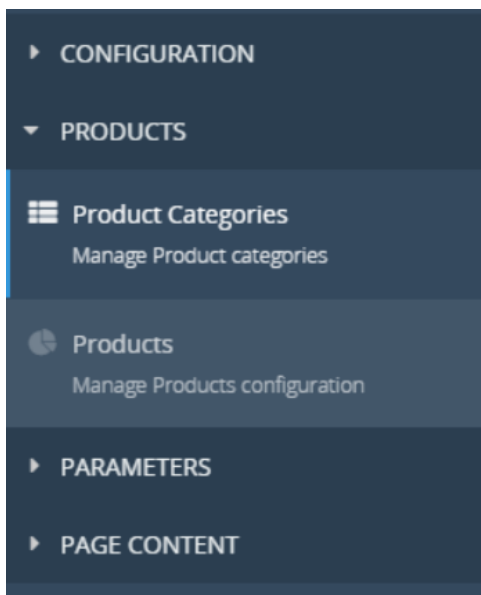
Description

A good protection for you and your family starts here.

Obrázok 20 Administrátorské rozhranie na nastavenie balíčkov.

4.1.2 Products

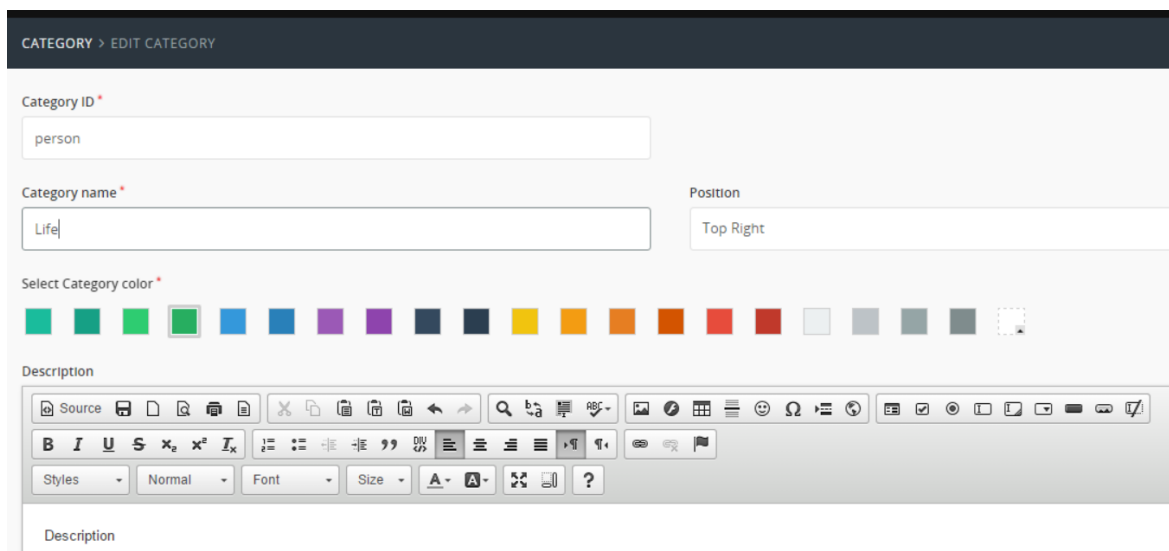
Táto kategória je zameraná na správu produktov a produktových kategórií.



Obrázok 21 Menu pre kategóriu Products.

4.1.2.1 Product Categories

Produktové kategórie nemajú v sebe zahrnutú žiadnu špeciálnu funkcionálnu. Všetky časti, ktoré sú nastavovateľné sú nakonfigurované len v YAML súbore.



Obrázok 22 Administrátorské rozhranie na nastavenie produktových kategórií.

4.1.2.2 Products

Správa produktov je rozdelená do 3 častí. V prvej časti sa nastavujú bežné parametre produktu ako sú: meno, box meno, kategória, popis a OE Product ID. Toto ID je referenčné ID ktoré je namapované na produkčný backend poisťovne. Druhou časťou je správa produktových parametrov pre daný produkt a tretiu časť tvoria tzv. marketingové záložky označené ako TAB 1-3.

Obrázok 23 Administrátorské rozhranie na nastavenie produktu.

Pri programovaní produktovej sekcie som narazil na dve komplikácie. Prvou bolo zamedzenie vytváraniu produktových parametrov ešte predtým než bol samotný produkt vytvorený. Bez tohto obmedzenia bolo možné vytvoriť produktový parameter, ktorý vlastne nebol priradený k žiadnemu produktu.

Na vyriešenie tohto problému som použil funkcionality, ktorú ponúka konfiguračný YAML súbor. Tam je možné jednoducho zadefinovať, ktoré polia sa zobrazia pri vytváraní nového záznamu, v našom prípade nového produktu a ktoré polia sa zobrazia len pri aktualizácii. Týmto spôsobom som pri vytváraní nového produktu zobrazil na mieste kde sa vytvárajú produktové parametre len informáciu o tom, kedy bude možné pridávať parametre.

```
64     parameters@create:  
65         label: Product Parameters  
66         type: section  
67         comment: If you want to add Product Parameters you must create Product.  
68         tab: Product Parameters  
69  
70     parameters@update:  
71         type: partial  
72         path: $/azl/allianzl/controllers/product/_field_parameters.htm  
73         tab: Product Parameters
```

Obrázok 24 Nakonfigurovanie zobrazenia položky parametrov v závislosti od spôsobu otvorenia.

Druhá komplikácia je spojená s vytváraním balíčkov v sekcii FastQuotation, kde sa označujú produkty, ktoré má daný balíček obsahovať. Problém nastal, keď sa produkt, ktorý bol označený v nejakom balíčku stal neaktívnym, alebo sa daný produkt odstránil.

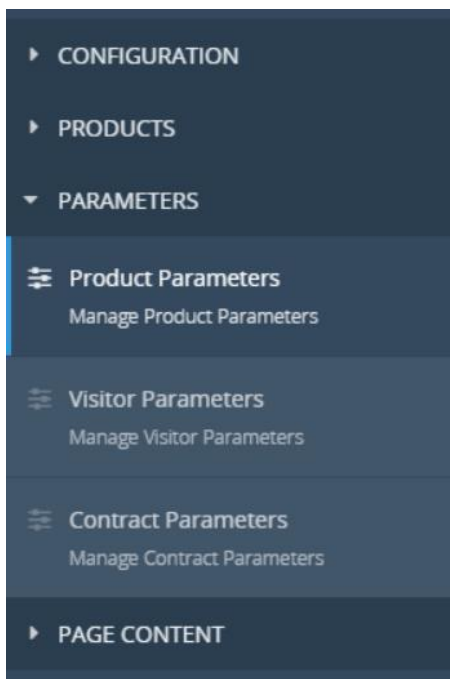
Pre takýto prípad je možné aplikovať funkciu **beforeSave**, ktorá už podľa názvu naznačuje, že je volaná pred uložením daného produktu. V tejto funkcii som si načítal list všetkých produktov použitých v balíčkoch a ak sa ID zhodovalo s produktom, ktorý sa práve stal neaktívnym tak sa daný produkt z toho balíčku odstránil.

```
127 public function beforeSave() {
128     if ($this->enabled == 0) {
129
130         $products = Fastquotation::all()->lists('required_products', 'id');
131
132         foreach ($products as $id => $product) {
133             if (!empty($product) && ($key = array_search($this->remote_id, $product)) !== false) {
134                 unset($product[$key]);
135                 Fastquotation::where('id', $id)->update(['required_products' => json_encode(array_values($product))]);
136             }
137         }
138     }
139 }
140
141 }
```

Obrázok 25 Funkcia na kontrolu neaktívnych produktov definovaných v balíčkoch.

4.1.3 Parameters

V celej aplikácii figurujú tri typy parametrov, ktoré sú z pohľadu nakonfigurovania v PCT veľmi podobné.



Obrázok 26 Rozdelenie menu pre kategóriu Parameters.

4.1.3.1 Product Parameters

Správa produktových parametrov je pre lepšiu orientáciu v administrácii prístupná na dvoch miestach. Prvé miesto bolo už spomenuté v popise produktov a druhé je v samostatnej kategórii pre správu parametrov. Rozdiel pri vytváraní produktového parametra je, že z prostredia administrácie produktu ma každý nový parameter už v sebe priradenú referenciu na produkt, ale zo sekcie Parameters je nutné zvoliť pre ktorý produkt je daný parameter priradený. Každý produktový parameter musí byť priradený k konkrétnemu produktu.

Hlavnú časť každého parametra tvorí jeho typ. Admin si môže vybrať zo staticky preddefinovaného zoznamu podporovaných typov.

```

48     type:
49         label: Type
50         span: left
51         type: dropdown
52         options:
53             text: Text
54             number: Number
55             checkbox: Checkbox
56             radio: Radio buttons
57             uiselect: Selectbox
58             slider: Slider
59             multiCheckbox: Multi-checkbox
60             datePicker: Date Picker

```

Obrázok 27 Preddefinovaný list typov pre produktový parameter

V závislosti od typu je ďalej aplikovaná logika, ktoré doplnkové polia sa dajú konfigurovať. Čiže je vytvorená závislosť medzi jednotlivými poľami.

```

76     public function filterFields($fields, $context = null)
77     {
78         $config = Config::All()->first()->toArray();
79
80         $nonRequiredType = ['checkbox', 'datePicker', 'number', 'slider'];
81
82         if (in_array($this->type, $nonRequiredType)) {
83             $fields->required->hidden = true;
84         }
85
86         if($config['remote_correlations']) {
87             $fields->correlation_id->hidden = true;
88         }
89
90         if ($this->type == 'text') {
91             $fields->pattern->hidden = false;
92         }
93
94         if ($this->type == 'slider') {
95             $fields->slider_step->hidden = false;
96             $fields->slider_min_max->hidden = false;
97         }
98
99         if ($this->type == 'uiselect') {
100             $fields->autocomplete->hidden = false;
101         } else {
102             $fields->datasource_id->hidden = true;
103         }
104
105         if ($this->datasource_id != '0') {
106             $fields->preload->hidden = false;
107         }
108
109         if ($this->datasource_id == 1) {
110             $fields->remote_api->hidden = false;
111         }
112
113         return $fields;
114     }

```

Obrázok 28 Funkcia, ktorá je automaticky volaná pri zmene hodnoty poľa realizovaná pomocou AJAX frameworku.

4.1.3.2 Visitor a Contract Parameters

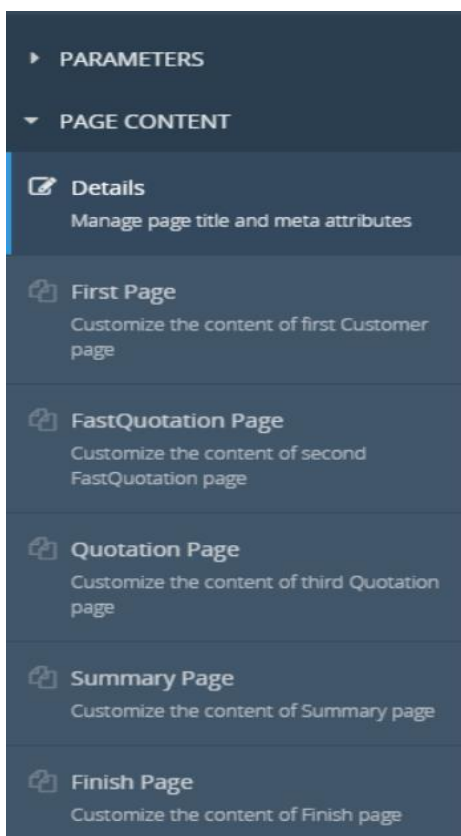
Majú rovnakú konfiguráciu ako produktové parametre, tým rozdielom, že nie sú referované k žiadnemu produktu.

4.1.4 Page Content

Poslednou časťou je správa webového obsahu jednotlivých stránok. Vytvorenie jednej zákaznickej ponuky prebieha postupne na 5 stránkach. Každá z týchto stránok je zložená z troch základných častí:

- **Hlavička:** kde administrátor môže zadať nadpis , alebo krátky informačný text.
- **Telo:** V tomto poli už je zahrnutý aj WYSIWYG editor, pre efektívnejšiu prácu s textom
- **Zapätie:** Rovnaká funkcionlita ako má telo.

Ostatné súčasti, ktoré môže administrátor upravovať sú pre každú stránku jedinečné.



Obrázok 29 Rozdelenie menu pre kategóriu Page Content.

4.1.4.1 Details

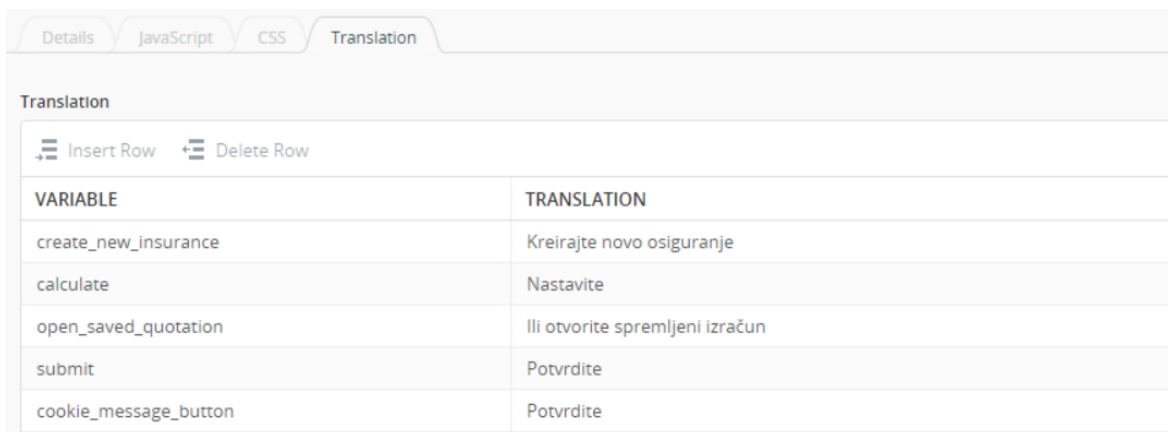
Na položke Details sa nastavujú spoločné atribúty pre všetky stránky. Ide o nastavenie titulu stránky, meta tagov, poprípade vlastné css alebo javascript kód, ktorý je možné zdefinovať pre záhlavie alebo zapätie stránky.

Keďže aplikácia je navrhnutá pre medzinárodné pobočky je nutné zahrnúť sekciu pre jazykové preklady tlačidiel, jednoduchých statických popisov a podobne. K tomuto účelu som použil typ **datatable**, ktorý predstavuje jednoduchú tabuľku. V nej som zdefinoval dva stĺpce.

```
64         translation:
65             label: Translation
66             type: datatable
67             tab: Translation
68             columns:
69                 variable:
70                     title: Variable
71                     width: 30%
72                 translation:
73                     title: Translation
```

Obrázok 30 Zedefinovanie poľa na preklad, pomocou typu datatable.

Vo výsledku bude mať admin možnosť pridávať a odoberať riadky tabuľky a podľa potreby a editovať existujúce záznamy.



The screenshot shows a web interface with tabs for 'Details', 'JavaScript', 'CSS', and 'Translation'. The 'Translation' tab is active, displaying a table with two columns: 'VARIABLE' and 'TRANSLATION'. The table contains six rows of data. Above the table, there are buttons for 'Insert Row' and 'Delete Row'.

VARIABLE	TRANSLATION
create_new_insurance	Kreirajte novo osiguranje
calculate	Nastavite
open_saved_quotation	Ili otvorite spremljeni izračun
submit	Potvrdite
cookie_message_button	Potvrdite

Obrázok 31 Ukážka administrácie na správu prekladov.

Keďže princíp multi-jazykovej stránky je založený na fyzickom prekladovom súbore, bolo nutné pred každým uložením prekladov aktualizovať takýto súbor.

```
69     public function beforeSave() {
70
71         $content = [];
72         foreach ($this->translation as $translation) {
73             $content[$translation['variable']] = $translation['translation'];
74         }
75
76         $directory = Storage::disk('orch')->put('translation.php', '<?php return '.var_export($content,true).');
77     }
```

Obrázok 32 Funkcia ktorá pred uložením aktualizuje súbor translation.php.

4.1.4.2 First Page

V sebe zahŕňa správu obsahu pre prvú stránku aplikácie. Okrem spoločných častí obsahuje prvá stránka video slideshow a sekciu pre krátke články. Obe tieto časti sú nakonfigurované ako typ **repeater** . V ktorom je možnosť nadefinovať skupiny polí rôznych typov.

```
29 customer_body:
30   label: Middle Content :
31   type: repeater
32   tab: Middle Content
33   prompt: Add new Content
34   form:
35     fields:
36       title:
37         label: Title
38         type: text
39       body:
40         label: Details
41         type: richeditor
```

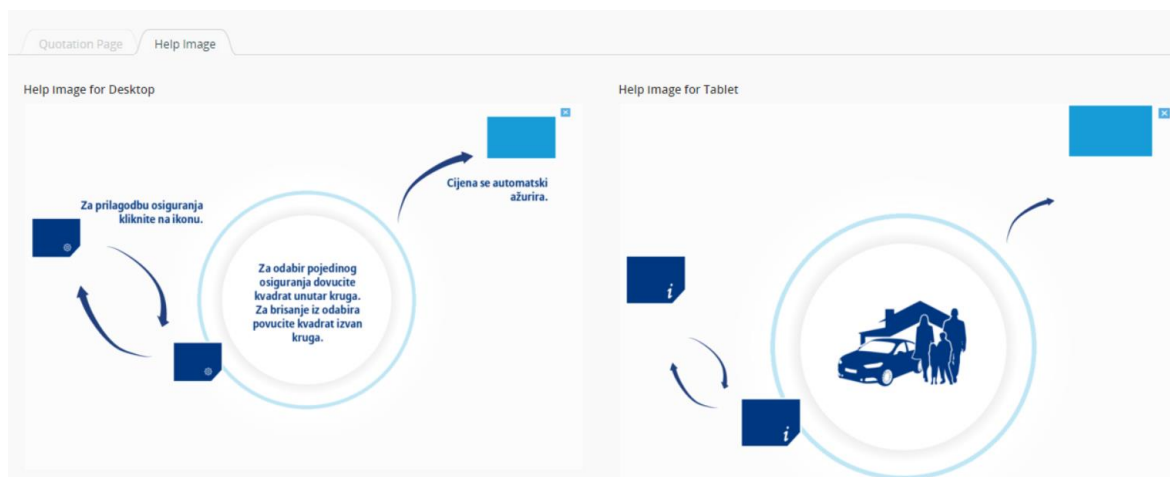
Obrázok 33 Konfigurácie časti pre krátke články

4.1.4.3 FastQuotation Page

Pre stránku balíčkov nie je nič jedinečné, takže obsahujú správu len spoločných častí.

4.1.4.4 Quotation Page

V tejto časti má administrátor možnosť vybrať tzv. pomocné obrázky, ktoré sa zobrazia zákazníkom pri vytvorení novej ponuky. Keďže je aplikácia podporovaná aj pre tablety a telefóny musí administrátor nahráť pre každé zariadenie iný obrázok s iným rozlíšením.



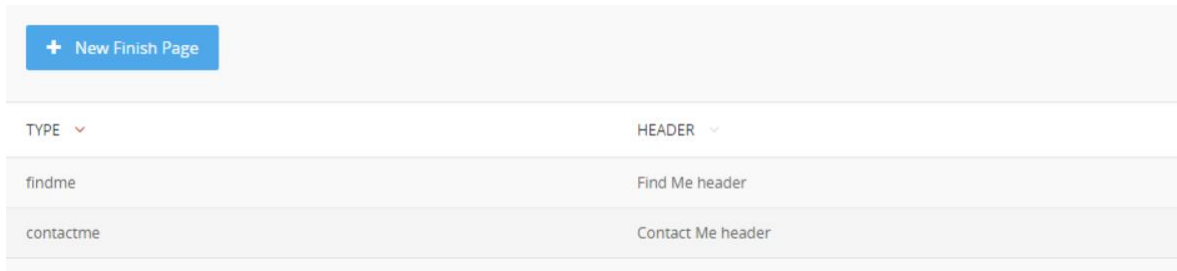
Obrázok 34 Časť administrácie na výber pomocných obrázkov.

4.1.4.5 Summary Page

Rovnako ako FastQuotation Page neobsahuje žiadnu špeciálnu funkcionality.

4.1.4.6 Finish Page

Predstavuje niečo ako ďakovné stránky. Podľa počtu metód ukončenia vytvorenej ponuky si administrátor môže nadefinovať svoje ďakovné texty.



TYPE	HEADER
findme	Find Me header
contactme	Contact Me header

Obrázok 35 Ukážka administrácie, kde sú vytvorené dve ďakovné stránky.

4.2 Exambler

Exambler je v podstate middleware aplikácie, kde je zahrnutá všetka logika. Jeho úlohou je spracovať a poskytnúť všetky potrebné informácie frontendu. Okrem frontendu komunikuje s databázou a produkčným backendom.

4.2.1 HTTP smerovanie

Všetky cesty sú definované v súbore **routes.php**, ktorý je automaticky načítaný frameworkom. Laravel poskytuje veľmi jednoduchý a expresívny spôsob vymedzenia trasy. Router umožňuje definovať všetky typy http volaní.

```
75 // quotations
76 Route::post('quotations', 'Api\QuotationsController@store');
77 Route::get('quotations/{quotID}', 'Api\QuotationsController@show');
78 Route::put('quotations/{quotID}', 'Api\QuotationsController@update');
```

Obrázok 36 Ukážka API cesty v súbore routes.php

Ja som si jednotlivé volania rozdelil do dvoch kategórii:

- Pre API volania
- Pre UI volania

Každé volanie je ďalej smerované do radiča na odpovedajúcu metódu. Každá metóda je zhruba zložená zo vstupnej časti, kde spracuje vstupné dáta. Znovu som vytvoril skupinu formátov teraz pre triedy API volaní. Po naformátovaní na požadovanú štruktúru sa dáta posielajú do brány, kde sa vykoná všetka logika pre dané volanie a nakoniec sa opäť pošlú dáta do výstupného formátora.

4.2.2 Brána

Všetka aplikačná logika je realizovaná na tomto mieste. Hlavnou úlohou je patričným spôsobom spracovať dáta ktoré prídu z troch zdrojov:

- Dáta z radiča
- Dáta z databázy
- Dáta z web servisov

4.2.3 Backend

Aplikácia ponúka dve komunikačné rozhrania, ktoré sú na základe nastavenia v PCT volané. Keďže aplikácia je ešte stále vo vývoji v prevádzke je momentálne len komunikácia prostredníctvom SOAP protokolu. K tomuto účelu som na základe špecifikácií web servisov vytvoril interface.

```
1 <?php namespace App\Backend;
2
3 interface BackendApiInterface {
4     public function getFastQuotations($requestData);
5     public function getLocations($search);
6     public function getProfessions($search);
7     public function getProductsForExistingUser($requestData, $listOfProducts);
8     public function getProductsForNewUser($requestData, $availableProducts, $requiredProducts);
9     public function getProductParameters($requestData);
10    public function recalculateProductPrices($requestData);
11    public function finishQuotation($requestData);
12 }
```

Obrázok 37 Backend interface pre SOAP komunikáciu

Druhým krokom bola implementácia SOAP klienta. Keďže framework Laravel ho v sebe nemá zahrnutý, použil som veľmi obľúbenú knižnicu nuSoap.

Pre použitie nuSoap klienta som vytvoril konštruktor, v ktorom sa nastaví všetky potrebné časti ako sú: koncový bod, kódovanie, proxy alebo zabezpečenie.

```
16     public function __construct() {
17
18         $endpoint = Config::get('backend.location');
19
20         $this->client = new \nuSOAP_client($endpoint . '?wsdl', 'wsdl');
21         $this->client->soap_defencoding = 'UTF-8';
22         $this->client->decode_utf8 = false;
23         $this->client->setEndpoint($endpoint);
24
25         if (Config::get('backend.ntlm')) {
26             $this->client->setCredentials(Config::get('backend.ntlm_name'), Config::get('backend.ntlm_password'), 'ntlm');
27         }
28
29         $this->client->setHTTPProxy(
30             Config::get('backend.proxy_host'),
31             Config::get('backend.proxy_port'),
32             Config::get('backend.proxy_login'),
33             Config::get('backend.proxy_password')
34         );
35     }
```

Obrázok 38 Konštruktor pre SOAP backend API

Všetky funkcie vytvorené v tejto triede potom môžu priamo používať klienta na volanie jednotlivých služieb.

Keďže Exambler predstavuje transformačnú vrstvu medzi frontendom aplikácie a produkčným backendom je nutné zachovať konzistenciu a štruktúru dát pre jednotlivé prostredia. K tomuto účelu som si vytvoril špeciálnu kategóriu formátarov, kde pre každé volanie mám výstupný formáter a pre odpoveď vstupný formáter.

```
259     public function getProductsForNewUser($data, $availableProducts, $requiredProducts) {
260         $formatter = new GetProductsForNewUserFormatter();
261         $data = $formatter->formatRequest($data, $availableProducts, $requiredProducts);
262     }
```

Obrázok 39 Ukážka kódu, kde sa všetky vstupné data posielajú do funkcie formátara, kde sa upraví na požadovanú štruktúru.

Každá odpoveď z klienta je ďalej kontrolovaná či nenastala v komunikácii chyba. Ak sa vyhodnotí chyba je vyvolaná výnimka, ktorá skontroluje chybový kód. Keďže v PCT môže administrátor definovať správy pre chybové kódy, tak sa najskôr prehľadá databáza či existuje preddefinovaná správa, ktorá by sa užívateľovi zobrazila ak sa nenájde použije sa základne chybové hlásenie.

```
269         if ($this->client->fault) {
270             $message = Handler::renderBackendException($response);
271
272             if (empty($message)) {
273                 $message = $soapFault->getMessage();
274             }
275
276             throw new BackendException(serialize($message), 1);
277         }
278     }
```

Obrázok 40 Ukážka kódu, kde sa zpracovávajú chyby zo SOAP klienta.

4.2.4 Repozitáre

V sebe zahŕňajú operácie nad databázou. Laravel poskytuje relatívne jednoduchú implementáciu **ActiveRecord** pre prácu s databázou. Každá databázová tabuľka má zodpovedajúci **Model**, ktorý sa používa pre komunikáciu s touto tabuľkou. Modely umožňujú volať na dáta v tabuľkách, rovnako ako vkladať nové záznamy do tabuliek. [10]

Podstata repozitára je zavolať príslušný typ volania do databázy. Vybrané dáta ďalej poslať do príslušného formátara a takto spracované dáta ďalej poskytnúť ďalšej vrstve aplikácie.

```

35 public function getVisitorParametersRemoteApi() {
36     $queryResults = VisitorParameter::where('enabled', 1)
37         ->where('remote_api', '!=', '')
38         ->select('remote_api')->get()->toArray();
39
40     $visitorParametersRemoteApi = $this->formatter->getVisitorParametersResponseFormatter($queryResults);
41
42     return $visitorParametersRemoteApi;
43 }

```

Obrázok 41 Ukážka funkcie pre výber dát z databázy, ich spracovanie a odoslanie ako výstup.

4.2.5 Služby

Pre každý formát je úlohou zoradiť, upraviť alebo inak preformátovať dáta aby mali vo výsledku požadovaný tvar a štruktúru. Keďže tieto formátovacie operácie sú prevažne podobné vytvoril som si špeciálnu kategóriu pomocných služieb, ktoré vykonávajú operácie nad dátami.

4.2.5.1 Pomocné funkcie

Medzi pomocné funkcie som zaradil prevažne operácie s poľami. Filtrovanie, vymazávanie alebo premenovávanie.

```

6 public static function whitelistFilter($array, $whitelist) {
7     return array_intersect_key($array, array_flip($whitelist));
8 }
9
10 public static function rename($array, $keys) {
11     foreach ($keys as $originalKey => $newKey) {
12
13         // Skip keys that doesn't exist
14         if (!array_key_exists($originalKey, $array)) {
15             continue;
16         }
17
18         $array[$newKey] = $array[$originalKey];
19         unset($array[$originalKey]);
20     }
21
22     return $array;
23 }
24
25 public static function remove($array, $keys) {
26     foreach ($keys as $key) {
27
28         // Skip keys that doesn't exist
29         if (!array_key_exists($key, $array)) {
30             continue;
31         }
32
33         unset($array[$key]);
34     }
35
36     return $array;
37 }

```

Obrázok 42 Ukážka pomocných funkcií na úpravu poľa.

4.2.5.2 Placeholders systém

Aplikácie v sebe nesie veľa textov a popisov ktoré sú zadefinované v PCT. Avšak v určitých prípadoch je nutné ponúknuť dynamické texty vychádzajúce z nastavenia zákazníka. Pre tento účel sa špecifikoval placeholders systém.

PCT admin môže použiť pri písaní popisov špeciálny zápis premenných, ktoré sa následne nahradia určenými hodnotami.

V špecifikácii web servisov je možné po každej re-kalkulácii poslať sadu placeholderov, ktoré budú použité na vygenerovanie týchto dynamických popisov.

```
401 <xsd:element name="placeholders" minOccurs="0" maxOccurs="unbounded">
402   <xsd:complexType>
403     <xsd:sequence>
404       <xsd:element type="xsd:string" name="search"/>
405       <xsd:element type="xsd:string" name="replace"/>
406     </xsd:sequence>
407   </xsd:complexType>
408 </xsd:element>
```

Obrázok 43 Placeholder špecifikácia pre XSD schému.

Princíp spočíva v tom, že v texte vyhládam všetky špeciálne premenné, ktoré sa rovnajú hodnote **search**, ktorú dostanem z backendu a ak je zhoda nahradím to hodnotou **replace**.

Systém je ešte rozšírený o to aby aj hodnota **replace** bola dynamická a tým pádom preberala nejakú hodnotu z produktu, ktorý je uložený len v databáze aplikácie. Na tento účel som zadefinoval štruktúru použitia.

Tabuľka 12 Štruktúra pre dynamickú replace hodnotu.

Typ parametra	Oddeľovač	ID parametra
PRODUCT	&&	1

Pre lepšie pochopenie príklad použitia:

```
2 $placeholders = [
3   {
4     "search" => "PRODUCT_1",
5     "replace" => "product$$1"
6   },
7   {
8     "search" => "PRODUCT_2",
9     "replace" => "product$$2"
10  }
11 ]
```

Obrázok 44 Pole objektov s dvomi placeholderami.

Za predpokladu že v databáze sú uložené dva produkty:

- id : 1 , meno : MENO1
- id : 2 , meno : MENO2

\$text = "Produkt – {PRODUCT_1} nie je možné mať spolu s – {PRODUCT_2}";

\$výsledok = "Produkt – MENO1 nie je možné mať spolu s – MENO2";

4.2.5.3 Update Systém

Keďže aplikácia pracuje s dátami, ktoré prichádzajú z troch zdrojov je potreba všetky tieto dáta patrične pospájať alebo aktualizovať v závislosti od zmien. Pre tento účel som si vytvoril samostatné funkcie. Okrem jednoduchého spájania dvoch objektov sa sem pripája funkcionálna korelácia.

Korelácia funguje na princípe, že administrátor môže v PCT označiť dva alebo viacej parametrov do jednej korelačnej skupiny. Na základe tejto skupiny budú mať všetky tieto parametre rovnakú hodnotu. Ide len o to, z ktorého parametra sa má táto hodnota prebrať a aktualizovať ňou ostatných. Túto informáciu dostávam ako súčasť požiadavky na prepočítanie ceny.

```

5      public static function updateParameters($originalParameters, $newParameters, $mergeKey = 'remoteId', $partialUpdate = false) {
6          $mergedParameters = [];
7
8          foreach ($originalParameters as $originalParameter) {
9
10             $foundMatch = false;
11
12             foreach ($newParameters as $newParameter) {
13                 if ($originalParameter[$mergeKey] == $newParameter[$mergeKey]) {
14
15                     $foundMatch = true;

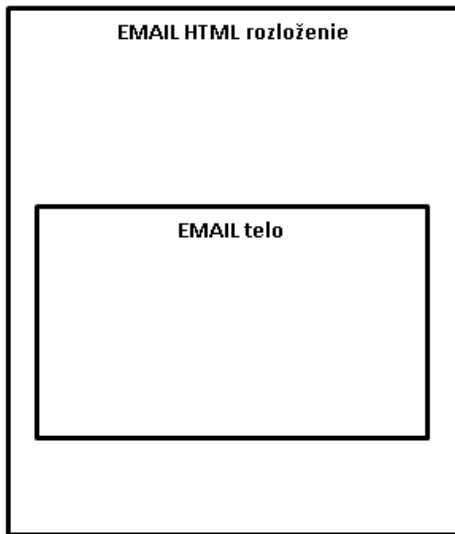
```

Obrázok 45 Časť funkcie na aktualizovanie parametrov.

4.2.5.4 Mail systém

Možnosťou admina je si v PCT nadefinovať vlastné emailové šablóny. Každá kompletná emailová šablóna je zložená z 2 častí:

- **HTML rozloženie:** Je to definícia základnej štruktúry emailu. Hlavička, zápätie, CSS štýly poprípade iné spoločné rysy pre email.
- **Telo:** Obsahuje telo emailu.



Obrázok 46 Zloženie emailovej šablóny.

Pre spojenie týchto dvoch častí som využil taktiež druh placeholdera. Na miesto **{templateBody}** je nahraný obsah z EMAIL tela.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Page Title</title>
5   </head>
6   <body>
7     {templateBody}
8   </body>
9 </html>

```

Obrázok 47 Základné zloženie pre HTML rozloženie.

Tabuľka 13 Príklad vytvorenia kompletnej emailovej šablóny

HTML ROZLOŽENIE	EMAIL TELO	VÝSLEDOK
<pre> <!DOCTYPE html> <html> <head> <title>Page Title</title> </head> <body> {templateBody} </body> </html> </pre>	<pre> <table> <tr><td>TEST</td><tr> </table> </pre>	<pre> <!DOCTYPE html> <html> <head> <title>Page Title</title> </head> <body> <table> <tr><td>TEST</td><tr> </table> </body> </html> </pre>

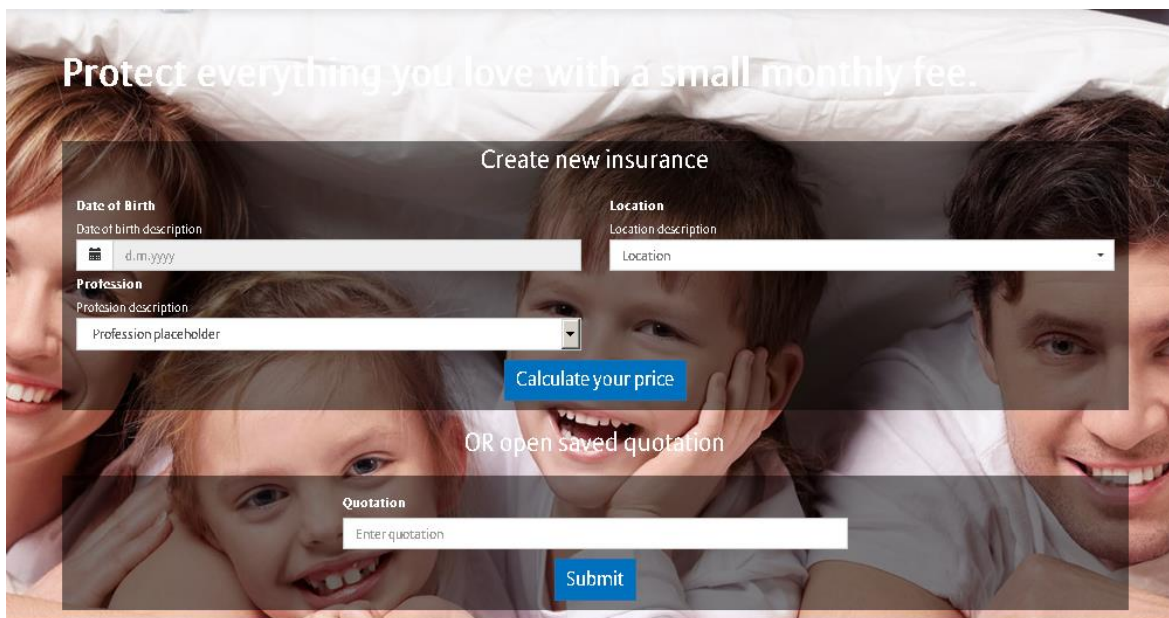
4.3 Frontend

Frontend aplikácie je založený na Blade engine od frameworku Laravel. Blade je jednoduchý, ale účinný šablónový systém. Na rozdiel od iných populárnych PHP šablónových enginov, neobmedzuje používanie obyčajného PHP kódu vo view častiach.

Keďže programovanie Frontend časti aplikácie nie je predmetom diplomovej práce, nepovažujem za nutné rozpisovať funkčnosť jednotlivých častí pre účely tejto práce.

4.3.1 Úvodná stránka

Na prvej stránke, návštevník zadáva tri vstupné údaje z ktorých sú vypočítané ceny balíčkov, alebo má možnosť si otvoriť už svoju existujúcu ponuku. Všetky tieto vstupy sú definovateľné v PCT a označované ako visitor parameters.



Obrázok 48 Náhľad na prvú stránku, kde návštevník zadáva vstupné údaje.

4.3.2 Stránka vypočítaných balíčkov

Na druhej stránke sa vygenerujú spomínané balíčky, z ktorých si užívateľ môže vybrať. Tieto balíčky sú definovateľné v PCT pod označením FastQuotation a pred vygenerovaním sa k nim ešte pripája cena ktorá je poskytovaná produkčným backendom.

How do you want to continue?

Blank fast quotation

Or start from a solution for you

Silver	Golden	Diamond	Bronze
You and your home is protected from the unexpected.	Safety for you and your family, inside and outside the home.	Protection for what you care about most.	A good protection for you and your family starts here. A good protection for y...
House: <input type="checkbox"/> Vehicle: <input type="checkbox"/> Life: <input type="checkbox"/>	House: <input type="checkbox"/> Vehicle: <input type="checkbox"/> Life: <input type="checkbox"/>	House: <input type="checkbox"/> Vehicle: <input type="checkbox"/> Life: <input type="checkbox"/>	House: <input type="checkbox"/> Vehicle: <input type="checkbox"/> Life: <input type="checkbox"/>
15 € per month	18 € per month	20 € per month	55 € per month

Prices are indicative only

Obrázok 49 Náhľad na druhú stránku, kde užívateľ môže vybrať z ponúkaných balíčkov.

4.3.3 Stránka ponuky

Na základe vybraného balíčka je vygenerovaná stránka ponuky, kde užívateľ vidí všetky dostupné produkty. Tie produkty, ktoré sú v kruhu tvoria ponuku poistenia. Pridávanie alebo odoberanie produktov z kruhu funguje na princípe Drag&Drop. Na vstupných parametroch môžu byť niektoré produkty zakázané

Help ?

Choose the combination of modules and customizable

HOUSE
Protect your home from the unexpected events

Personal estate insurance	9 €
Assistance insurance	8 €

VEHICLE
Service for your car

Accident daily change insurance	7 €
---------------------------------	-----

Personal insurance House 8 €

Building insurance House 10 €

Accidental disability insurance Vehicle 7 €

Life Insurance V 3 €

Life Insurance III 2 €

30 €
PER MONTH

LIFE

Take care of yourself and the people you love.

Life Insurance VI	4 €
Life Insurance IV	3 €
Life Insurance II	2 €

Switch quotation + Add all products

Test footer

QUOTATION SUMMARY 1 >

Obrázok 50 Náhľad na stránku ponuky, kde užívateľ môže pridávať a odoberať produkty.

4.3.4 Konfigurácia produktu

Po kliknutí na znak ikonky konfigurácie pri produkte má užívateľ možnosť tento produkt ešte upraviť. Načítajú sa pre daný produkt jeho produktové parametre, ktoré sú definované v PCT.

The screenshot shows a configuration page for 'Building insurance'. At the top right, there is a price tag: '10 € per month' with a close icon. Below the title, a description reads: 'Protects you from damage caused to others by you, by the members of your household or domestic helpers in everyday life.' The configuration is divided into three tabs: 'Customize' (active), 'Service', and 'Video'. Under 'Customize', there are several sections: 'Number' with a text input containing '4'; 'Select' with a dropdown menu showing 'select1'; 'Permanent disability' with radio buttons for 'flat', 'house', and 'tent'; and 'Price X' with a slider ranging from 5.000 to 10.000, currently set at 2. On the right side, under the 'Service' tab, there are sections for 'Email' (with a text description input), 'checkbox' (with a checked checkbox), and 'Multicheckbox' (with checkboxes for 'house' and 'Building', where 'Building' is checked). At the bottom, there are two buttons: 'Cancel changes' and 'Confirm'.

Obrázok 51 Náhľad na konfiguráciu produktových parametrov.

4.3.5 Sumarizácia ponuky

Po dokončení všetkých úprav na ponuke sa na ďalšej stránke zobrazí sumár vybraných produktov a popis k tomu. Aj na tejto stránke môže užívateľ ešte konfigurovať produkty alebo ich odstrániť z ponuky.

1 Here is your quotation ID:
ADHAWA

Click on the button to find your nearest buying agency.

2 Find agency location:

Use to recover at any time your solution and take it to the agency.

Search agency

Your solution

Additional body text

		Contact me	22 € <small>per month</small>				
	<p>House Protect your home from the unexpected events</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%; text-align: center; padding: 5px;">Building insurance</td> <td style="width: 40%; padding: 5px;">Product summary description text.</td> <td style="width: 15%; text-align: center; padding: 5px;">⚙️ 🗑️</td> <td style="width: 15%; text-align: center; padding: 5px;">5 €</td> </tr> </table>	Building insurance	Product summary description text.	⚙️ 🗑️	5 €		
Building insurance	Product summary description text.	⚙️ 🗑️	5 €				
	<p>Life Take care of yourself and the people you love.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%; text-align: center; padding: 5px;">Life Insurance V</td> <td style="width: 40%; padding: 5px;"></td> <td style="width: 15%; text-align: center; padding: 5px;">⚙️ 🗑️</td> <td style="width: 15%; text-align: center; padding: 5px;">17 €</td> </tr> </table>	Life Insurance V		⚙️ 🗑️	17 €		
Life Insurance V		⚙️ 🗑️	17 €				

Back

Contact me

Footer for summary page

Obrázok 52 Náhľad na sumár vytvorenej ponuky.

4.3.6 Dokončenie ponuky

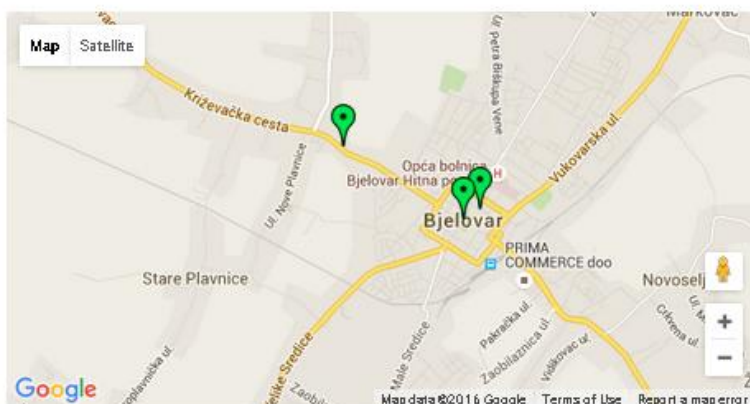
Ukončiť celý proces je možné dvomi spôsobmi.

- Prvý spôsob je po kliknutí na tlačidlo **Search Agency**. V tejto možnosti sa užívateľovi vygeneruje mapa s miestami najbližších agentov k jeho bydlisku, ktoré zadal na prvej stránke. Po kliknutí na načítaného agenta má možnosť vidieť jeho kontaktné údaje, kde si môže ponuku prísť podpísať.
- Druhý spôsob je tlačidlo **Contact me**. V tejto možnosti sa užívateľovi vygenerujú formuláre, ktoré vyplní, a príslušný agent ho tak môže v čo najkratšom čase kontaktovať.

Find me



Find me description



Osiguranje - Zastupstvo Bjelovar
F. Supila 8

Osiguranje - Prodajno Mjesto Bjelovar
Velike Sredice bb

Osiguranje - Prodajno Mjesto Bjelovar - Curoduhan
Križevačka cesta 2

Chosen agency details:

Osiguranje - Zastupstvo Bjelovar

F. Supila 8, 43000 BJELOVAR

Tel: 043 123 456 / Fax: 043 123 456

Email: example@email.xx

Location

Gradec

Email address

Enter your email

Submit

Obrázok 53 Náhl'ad na spôsob ukončenia, vyhľadat' agentov.

Contact me



Contact me description

First Name

Example block-level help text here

Enter your first name

Last Name

Example block-level help text here

Enter your last name

Phone

Example block-level help text here

Enter your phone

Email address

Example block-level help text here

Enter your email

Location

Example block-level help text here

Gradec

Submit

Obrázok 54 Náhl'ad na spôsob ukončenia, nechat' sa kontaktovat'.

ZÁVĚR

Cieľom diplomovej práce bolo navrhnúť a vytvoriť systém, ktorý nielen uľahčí prácu predajcov, ale najmä zaujme zákazníkov svojou atraktívnosťou a jednoduchosťou. Aplikácia pre zjednodušenie poistenia cez internet je v konečnom dôsledku mocný dynamický nástroj, ktorý je možné aplikovať pre najrôznejšie produktové balíčky poistenia.

Prvá časť práce bola zameraná na oboznámenie sa so základnými použitými vývojovými technológiami aplikácie. Pozornosť bola venovaná nielen samotným programovacím jazykom z ktorých je aplikácia naprogramovaná, ale taktiež programom, ktoré uľahčujú a organizujú celý proces vývoja.

Keďže využitie aplikácie má byť pre medzinárodné pobočky, bolo veľkou prioritou zozbieranie a zanalyzovanie všetkých požiadaviek. Požiadavky boli rozdelené do troch kategórií. Prvá kategória bola zameraná na požiadavky návštevníkov a zákazníkov, druhá na požiadavky aplikácie a tretiu najväčšiu časť tvoria požiadavky na celý konfiguračný nástroj. Na základe požiadavkou bola spísaná funkčná špecifikácia aplikácie, ktorej účelom je poskytnúť prehľad hlavných funkcií systému. Zahŕňa v sebe popis a architektúru aplikácie.

Implementačná časť je taktiež rozdelená na tri časti. Prvú časť tvorí konfiguračný nástroj PCT, ktorý má ponúknuť školeným administrátorom jednoduchú a rýchlu úpravu obsahu aplikácie. V tejto časti je priblížený postup vývoja všetkých dostupných funkcií, kde detailnejšiu časť predstavujú problematickejšie oblasti. Druhú časť tvorí Exambler, ktorý v sebe zahŕňa celú logickú časť aplikácie. A posledná časť zobrazuje finálnu podobu aplikácie.

Výsledkom tohto projektu je aplikácia ktorá znamená prínos v oblasti uzatvárania poistenia prostredníctvom internetu. Snaha o jednoduché a atraktívne ovládanie umožňuje takmer každému rýchlo a bez zbytočného čakania uzatvoriť poistenie.

Vývoj tejto aplikácie sa neustále rozvíja, nachádzame stále nové požiadavky a možnosti, ktoré by boli vhodné, alebo priam nutné zahrnúť.

SEZNAM POUŽITÉ LITERATURY

- [1] PHP-základy. *Tvorba-webu* [online]. [cit. 2016-05-17]. Dostupné z: <http://www.tvorba-webu.cz/php/>
- [2] General Information. *PHP* [online]. [cit. 2016-05-17]. Dostupné z: <http://php.net/manual/en/faq.general.php>
- [3] PHP (skriptovací jazyk), Wikipedia [online]. [cit. 2016-05-17]. Dostupné z: [https://sk.wikipedia.org/wiki/PHP_\(skriptovac%C3%AD_jazyk\)](https://sk.wikipedia.org/wiki/PHP_(skriptovac%C3%AD_jazyk))
- [4] PHP má 20 rokov, Zive [online]. [cit. 2016-05-17]. Dostupné z: <http://www.zive.sk/clanok/105621/php-ma-20-rokov-pozrite-si-milniky-ktore-mu-prinasaju-dominanciu>
- [5] Laravel Introduction, Laravel [online]. [cit. 2016-05-17]. Dostupné z: <https://laravel.com/docs/4.2/introduction>
- [6] Angular JS, Angularjs [online]. [cit. 2016-05-17]. Dostupné z: <https://angularjs.org/>
- [7] Lednár, Matej.: Príručka programátora - Prehľadný sprievodca jazykom CSS 2.1. Bratislava: MLD Group, 2009, s.148. ISBN 978-80-89448-03-6
- [8] MySQL, Wikipedia [online]. [cit. 2016-05-17]. Dostupné z: <https://sk.wikipedia.org/wiki/MySQL>
- [9] Git - Historie a principy, Itnetwork [online]. [cit. 2016-05-17]. Dostupné z: <http://www.itnetwork.cz/software/git/git-tutorial-historie-a-principy>
- [10] Laravel Eloquent, Laravel [online]. [cit. 2016-05-17]. Dostupné z: <https://laravel.com/docs/5.2/eloquent>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PCT	Product Configuration Tool
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language
RDC	Remote Desktop Connection
LAN	Local area network
MPLS	Multiprotocol Label Switching
OS	Operačný systém
CEE	Central and Eastern Europe
OE	Označenie partnerskej krajiny
HW	Hardware
CPU	Central processing unit
DEV	Developerské prostredie
UAT	User Acceptance Testing
PROD	Produkčné prostredie
WSDL	Web Services Description Language
CMS	Content management system
MVC	Model–view–controller
WYSIWYG	Online editor: What You See Is What You Get
API	Application programming interface
UI	User interface

SEZNAM OBRÁZKŮ

Obrázok 1 Logické súčasti aplikácie a ich interakcie.....	23
Obrázok 2 Fyzicko-aplikačná architektúra.....	25
Obrázok 3 Architektúra infraštruktúry	26
Obrázok 4 Aktéri aplikácie.....	32
Obrázok 5 Diagram prípadu užitia na online poistenie	33
Obrázok 6 Diagram prípadu užitia na proces ukončenia ponuky.....	36
Obrázok 7 Diagram prípadov užitia pre Administrátorov	38
Obrázok 8 Prieskum najobľúbenejších PHP Frameworkov v roku 2015.....	44
Obrázok 9 Adresárová štruktúra pluginov.....	45
Obrázok 10 Štruktúra registračného súboru Plugin.php.....	46
Obrázok 11 Základná štruktúra pre controller.....	46
Obrázok 12 Rozdelenie položiek menu do logických skupín.....	47
Obrázok 13 Menu pre kategóriu Configuration.....	47
Obrázok 14 YAML konfigurácia pre sekciu Web Services.....	48
Obrázok 15 Výsledok po nakonfigurovaní.....	48
Obrázok 16 Funkcia načítania všetkých parametrov, ktoré používajú vytvorenú koreláciu.....	49
Obrázok 17 Funkcia pre rýchle odstránenie korelácie z parametra.....	49
Obrázok 18 Administrácia korelácie, kde je vidieť zoznam všetkých parametrov ktoré majú v sebe priradenú Number koreláciu.....	49
Obrázok 19 Hodnoty sú kódované ako JSON pred uložením a následne prevedené na pole po načítaní.....	50
Obrázok 20 Administrátorské rozhranie na nastavenie balíčkov.....	50
Obrázok 21 Menu pre kategóriu Products.....	51
Obrázok 22 Administrátorské rozhranie na nastavenie produktových kategórií.....	51
Obrázok 23 Administrátorské rozhranie na nastavenie produktu.....	52
Obrázok 24 Nakonfigurovanie zobrazenia položky parametrov v závislosti od spôsobu otvorenia.....	52
Obrázok 25 Funkcia na kontrolu neaktívnych produktov definovaných v balíčkoch.....	53
Obrázok 26 Rozdelenie menu pre kategóriu Parameters.....	54
Obrázok 27 Preddefinovaný list typov pre produktový parameter.....	55

Obrázok 28 Funkcia, ktorá je automaticky volaná pri zmene hodnoty poľa realizovaná pomocou AJAX frameworku.	55
Obrázok 29 Rozdelenie menu pre kategóriu Page Content.	56
Obrázok 30 Zadefinovanie poľa na preklad, pomocou typu datatable.	57
Obrázok 31 Ukážka administrácie na správu prekladov.	57
Obrázok 32 Funkcia ktorá pred uložením aktualizuje súbor translation.php.	57
Obrázok 33 Konfigurácie časti pre krátke články	58
Obrázok 34 Časť administrácie na výber pomocných obrázkov.	58
Obrázok 35 Ukážka administrácie, kde sú vytvorené dve ďakovné stránky.	59
Obrázok 36 Ukážka API cesty v súbore routes.php	59
Obrázok 37 Backend interface pre SOAP komunikáciu	60
Obrázok 38 Konštruktor pre SOAP backend API	60
Obrázok 39 Ukážka kódu, kde sa všetky vstupné data posielajú do funkcie formátera, kde sa upravujú na požadovanú štruktúru.	61
Obrázok 40 Ukážka kódu, kde sa zpracovávajú chyby zo SOAP klienta.	61
Obrázok 41 Ukážka funkcie pre výber dát z databázy, ich spracovanie a odoslanie ako výstup.	62
Obrázok 42 Ukážka pomocných funkcií na úpravu poľa.	62
Obrázok 43 Placeholder špecifikácia pre XSD schému.	63
Obrázok 44 Pole objektov s dvomi placeholderami.	63
Obrázok 45 Časť funkcie na aktualizovanie parametrov.	64
Obrázok 46 Zloženie emailovej šablóny.	65
Obrázok 47 Základné zloženie pre HTML rozloženie.	65
Obrázok 48 Náhľad na prvú stránku, kde návštevník zadáva vstupné údaje.	66
Obrázok 49 Náhľad na druhú stránku, kde užívateľ môže vybrať z ponúkaných balíčkov.	67
Obrázok 50 Náhľad na stránku ponuky, kde užívateľ môže pridávať a odoberať produkty.	67
Obrázok 51 Náhľad na konfiguráciu produktových parametrov.	68
Obrázok 52 Náhľad na sumár vytvorenej ponuky.	69
Obrázok 53 Náhľad na spôsob ukončenia, vyhľadať agentov.	70
Obrázok 54 Náhľad na spôsob ukončenia, nechať sa kontaktovať.	70

SEZNAM TABULEK

Tabuľka 1 Doby odozvy systému	19
Tabuľka 2 Očakávaná priemerná doba odozvy pre hlavné užívateľské interakcie.	19
Tabuľka 3 Popis-logicko aplikačnej architektúri.....	23
Tabuľka 4 Vstupné parametre pre volanie getFastQuotations	27
Tabuľka 5 Výstupné parametre pre volanie getFastQuotations	28
Tabuľka 6 Vstupné parametre pre volanie getProductNewUser	28
Tabuľka 7 Výstupné parametre pre volanie getProductNewUser	29
Tabuľka 8 Vstupné parametre pre volanie getProductParameters.....	29
Tabuľka 9 Výstupné parametre pre volanie getProductParameters.....	29
Tabuľka 10 Vstupné parametre pre volanie recalculateProductPrice.....	30
Tabuľka 11 Výstupné parametre pre volanie recalculateProductPrice.....	30
Tabuľka 12 Štruktúra pre dynamickú replace hodnotu.	63
Tabuľka 13 Príklad vytvorenia kompletnej emailovej šablóny	65