Mendelova univerzita v Brně
Provozně ekonomická fakulta

Webová aplikace a administrační systém pro strojírenskou firmu

Diplomová práce

Vedoucí práce:
Ing. Jiří Lýsek, Ph.D.

Bc. Petr Janulík

Brno 2017

Zde bude přední strana zadání

Zde bude zadní strana zadání

Chtěl bych tímto způsobem poděkovat Ing. Jiřímu Lýskovi, Ph.D. za několik užitečných rad poskytnutých při psaní této práce. Především bych ale rád poděkoval celé své rodině, za bezmeznou podporu, které se mi od ní dostávalo po celou dobu studia.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Webová aplikace a administrační systém pro strojírenskou firmu**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 3. ledna 2017 .....

Abstract

Janulík, P. Web application and administration system for engineering company. Diploma thesis. Brno, 2017.

This diploma thesis deals with designing and implementation of administration system for Metal Ryspol, s. r. o. In the implementation phase, modern development technologies were used, such as MVC architecture and responsive web design.

Keywords

Administration system, MVC architecture, Laravel framework, Bootstrap, responsive design, Three.js

Abstrakt

Janulík, P. Webová aplikace a administrační systém pro strojírenskou firmu. Diplomová práce. Brno, 2017.

Diplomová práce se zabývá návrhem a implementací administračního systému a webové aplikace pro firmu Metal Ryspol, s. r. o. Při implementaci jsou použity moderní vývojářské technologie jako jsou MVC architektura či responzivní design.

Klíčová slova

Administrační systém, MVC architektura, Laravel framework, Bootstrap, responzivní design, Three.js.

Obsah

1 Úvod a cíl práce

1.1 Úvod do problematiky

V dnešní době internetu se považuje již za samozřejmé, že firma bude mít své webové stránky. Ve firemní webové prezentaci se může skrývat velký potenciál – záleží ovšem, jak se ke svému webu daná firma postaví. Mnoho podnikatelů svou firmu prezentuje na internetu jen proto, že se to od nich očekává. V takovém případě web většinou slouží pouze jako ověření, že firma existuje, „nějak vypadá“ a zákazník si tam jde maximálně tak pro kontakt. Přínosy plynoucí z takového webu jsou minimální.

Na druhou stranu je zde stále početnější skupina firem, které již zjistili, že vytvoření a provozování kvalitní internetové prezentace, výrazně přispívá k jejich prosperitě a zvyšuje jejich zisk a že v době, kdy je stále těžší a finančně náročnější prosadit se proti sílící konkurenci, může být nakonec kvalita firemního webu tím rozhodujícím faktorem.

Vybudování úspěšného firemního webu není však jednorázovou činností, ale jedná se o dlouhodobou ustavičnou činnost a je nutné počítat s tím, že to bude stát hodně času a také peněz. O web je potřeba pečovat již od samotného počátku jeho životního cyklu. Velký důraz je kladen především na obsah, který by měl být vždy aktuální. K tomuto účelu je zapotřebí kromě samotného webu také kvalitní systém pro správu obsahu – CMS (z anglického Content Management System). Díky CMS přestává platit to, že se o internetové stránky musí starat osoba s rozsáhlými znalostmi programování. Při používání CMS tvoří obsah stránek editoři, kteří nemusí vědět vůbec nic o tom, jak se webové stránky vytváří.

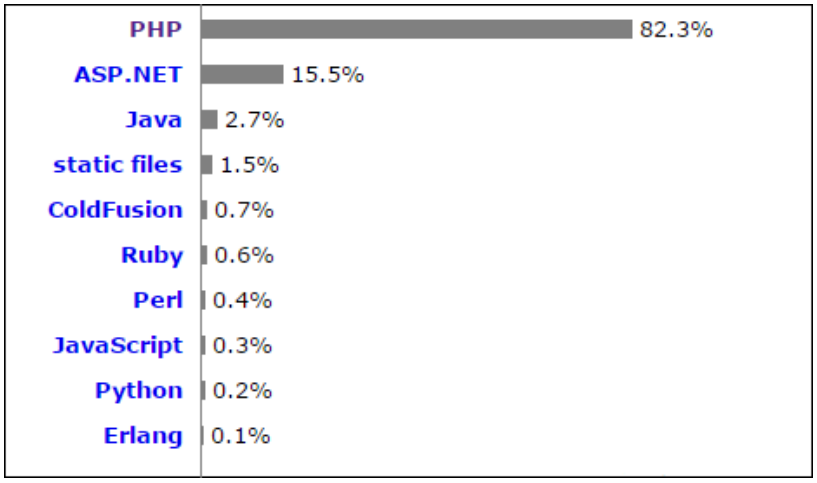
1.2 Cíl práce

Cílem této práce je návrh a implementace webové prezentace a administračního systému pro firmu Metal Ryspol, s. r. o. Dosažení tohoto cíle zahrnuje v první fázi provedení podrobné analýzy požadavků firmy na konečnou podobu webu a administrace, poté navržení vhodného řešení na základě předchozí analýzy a v neposlední řadě také implementaci tohoto návrhu. Na závěr bude implementované řešení otestováno a bude vyhodnocena míra splnění daného cíle, popřípadě budou navrženy možná vylepšení do budoucna.

2 Jazyk PHP a architektura MVC

PHP je programovací jazyk, který byl původně vytvořen Rasmusem Lerdorfem v roce 1994. Od této doby jazyk prošel značným vývojem a v současné době je dostupný již v sedmé verzi (History of PHP, 2016). PHP je navržen primárně pro vývoj webových aplikací, je však využíván také pro všeobecné účely. Jedná se o skriptovací jazyk (podobně jako například JavaScript), ale narozdíl právě třeba od JavaScriptu se nezpracovává v prohlížeči (ve vetšině případů), nýbrž na serveru, kde jsou webové stránky uloženy. Výhodou toho je, že ať už má člověk jakýkoliv prohlížeč, vždycky bude výsledek stejný. Tento princip přináší však nevýhodu v podobě nízké interaktivity, jelikož ke zpracování dojde až po odeslání požadavku na server.

PHP je v současné době (listopad 2016) nejrozšířenějším programovacím jazykem na straně serveru pro tvorbu webových stránek. S podílem přibližně 82 % má v tomto ohledu prakticky dominantní postavení (viz obrázek č. 1). Co se týče použití jednotlivých verzí, tak s přehledem stále vévodí PHP ve verzi 5 (97 %). Nejnovější sedmá verze se nachází na druhém místě s podílem pouhých 1,7 %.



Obrázek 1: Procentuální podíl webů využívající daný server-side programovací jazyk (Usage of server-side programming languages for websites, 2016)

Za touto popularitou stojí hned několik vlivů. Jedním z nich je jednoduchost a snadná naučitelnost jazyka. Dále jednoduchá instalace na sdílený hosting. Klíčovým vlivem toho, že se PHP v dnešní době stále těší takové popularitě bylo však vydání verze číslo 5. Ačkoliv tato aktualizace byla ve svých počátečních verzích doprovázena celou řadou problémů a skepsí programátorů zvyklých na PHP 4, tak i přesto by si dnes asi již mnoho lidí neumělo svoji práci bez pětky vůbec představit. Právě tato verze přinesla totiž možnosti pro vznik velkých nástrojů jako jsou Zend Framework, CakePHP, Symfony, Nette a další. A to hlavně svým vylepšeným objektovým přístupem, lepší podporou UTF-8, množstvím nových funkcí a knihoven

do PHP přímo zabudovaných, PDO knihovnou pro přístup k databázi a také solidní prací s XML (Přehled a vývoj PHP frameworků, 2008).

2.1 PHP frameworky

Framework lze definovat jako sadu knihoven, které v obecné podobě implementují časté programátorské konstrukce a usnadňují tak vývojářům rutinní práci. Bývají zaměřeny na určitou vymezenou oblast, např. tvorbu webových aplikací. Často jsou v nich aplikovány různé návrhové vzory a tzv. best practices z dané oblasti (Terminologický slovník, 2009).

Díky frameworku nemusíme vyvíjet webovou aplikací tzv. „z prázdného listu“. Tento přístup je v současné době již zastaralý, neefektivní a nese s sebou spoustu možných rizik a problémů. Mezi tyto problémy patří především složitější rozšiřitelnost již napsané aplikace. Tento přístup je dále velmi časově neefektivní, jelikož je vývojář nucen řešit notoricky známé problémy neustále dokola (například validace formulářů), což výrazně prodlužuje dobu vývoje.

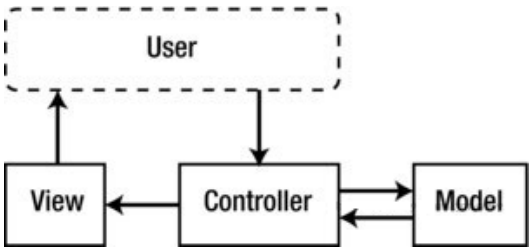
Právě tyto neduhy odstraňují frameworky, které usnadňují vývojářům život tím, že řeší typické problémy za něj a ten se tak může soustředit na jádro svého zadání. Dále je potřeba zmínit fakt, že tyto řešení jsou většiou velmi odladěné a vyskytuje se v nich minimum chyb. To je dáno tím, že jsou spravovány širokou základnou vývojářů, kteří se o své frameworky starají jako o vlastní dítě. Opravují chyby, přidávají nové funkce podle vlastních nápadů a realizují zajímavé a užitečné návrhy lidí, kteří jejich produkt používají.

V neposlední řadě jak již bylo zmíněno, frameworky mají v sobě velmi často zapracovány návrhové vzory a best practices z daných oblastí. Díky tomu jsou vývojáři nepřímo nuceni těchto praktik využívat, což vede jednak k dobrým programovacím návykům, ale také k velmi kvalitnímu a dobře udržovatelnému kódu. Typickým příkladem je použití architektury MVC, na které jsou postaveny nejlepší dnešní PHP frameworky.

2.2 Architektura MVC

Základní myšlenkou architektury MVC je oddělení logické části aplikace od výstupu, což přináší řešení pro klasický problém tzv. „špagetového kódu“, kdy máme v jednom souboru logické operace a zároveň renderování výstupu. Takový Soubor pak obsahuje databázové dotazy, logiku (např. PHP operace) a různě poházené HTML tagy – vše je zamotané do sebe jako špagety. To samozřejmě vede k velmi nepřehlednému a špatně udržovatelnému natož rozšiřitelnému kódu. Naším cílem je, aby zdrojový kód s logikou vypadal jako zdrojový kód (PHP) a výstup vypadal jako HTML stránka s co nejmenší příměsí dalšího kódu.

Architektura MVC řeší tento problém rozdělením aplikace do komponent tří typů. Hovoříme o modelech, pohledech a kontrolerech (Model-View-Controller). Schéma architektury je znázorněno na obrázku č. 2.



Obrázek 2: Schéma architektury MVC (Pro PHP MVC, 2012)

Jednotlivé komponenty

Pokud se podíváme na jednotlivé komponenty podrobněji, pak model představuje místo, kde je uchovávána veškerá business logika aplikace. Mohou to být výpočty, databázové dotazy, validace a podobně. Model vůbec neví o výstupu. Jeho funkce spočívá v přijetí parametrů zvenku a vydání dat ven. Pokud tedy nastane situace kdy je v aplikaci potřeba přistoupit k datům uloženým v databázi, pak kód, který obstarává tuto operaci, by měl být umístěn v modelu.

Pohled (View) se stará o zobrazení výstupu uživateli. Nejčastěji se jedná o šablonu, obsahující HTML stránku a tagy nějakého značkovacího jazyka, který umožňuje do šablony vkládat proměnné, případně provádět iterace (cykly) a podmínky. Dále pohledy mohou obsahovat CSS styly a JavaScriptové soubory. Zkrátka vše co koncový uživatel vidí na displeji svého zařízení nebo cokoliv s čím může interagovat, patří právě do pohledu.

Poslední komponentou je kontroler (Controller), který slouží jako prostředník se kterým komunikuje uživatel, model i pohled. Drží tedy celý systém pohromadě a komponenty propojuje v jeden celek (Pro PHP MVC, 2012).

Životní cyklus stránky

Princip toho jak jednotlivé komponenty vzájemně spolupracují a tvoří jeden celek, je nejlépe patrný na ukázce životního cyklu libovolné webové stránky. Cyklus je zahájen uživatelem, který zadá do svého webového prohlížeče adresu požadované stránky s parametry. Tento požadavek odešle na server. Příkladem takového požadavku může být následující URL adresa:

```
1 | http://www.domena.cz/produkt/detail/42
```

Na serveru nejprve požadavek odchytí tzv. router, který na základě prvního parametru URL adresy (produkt) pozná, jaký kontroler se má zavolat. V tomto případě bude zavolán ProduktController, který již přebere vyřizení požadavku do své režie. Kontroler podle zbylých dvou parametrů (detail a 42) pozná co se po něm požaduje. Z výše uvedené ukázky vyplývá, že má být zobrazena stránka, kde budou vypsány informace o produktu s identifikačním číslem 42.

Kontroler tedy nejprve zavolá příslušný model a požadá jej o informace o produktu číslo 42. Model tento požadavek přijme a zpracuje tak, že vyhledá v databázi

požadovaná data a nakonec je předá kontroleru, který si je ukladá do svých proměnných. Těchto požadavků kontroleru směrem k modelům může být více.

Poté co kontroler získá všechna potřebná data, předá je pohledu. Pohled sestaví výslednou HTML stránku tím, že vloží předaná data do požadované šablony. Hotová stránka je pak poslána uživateli jako odpověď serveru na jeho požadavek. Tato odpověď je v posledním kroku zpracována webovým prohlížečem, který stránku zobrazí na displeji koncového uživatele.

2.3 Přehled PHP frameworků

V následujícím textu je z široké nabídky frameworků vybráno a popsáno pět z nich. Výběr je založen na výsledcích ankety serveru Sitepoint.com (SitePoint Survey Results, 2015). V této anketě hlasovalo téměř 8 000 respondentů z celého světa, kde jejich úkolem bylo zvolit nejpopulárnější PHP framework. Tento průzkum může být do jisté míry zkreslený, jelikož je patrné, že jeho výsledky jsou až příliš závislé na demografickém rozdělení respondentů.

Dobrým příkladem tohoto jevu je celkové umístění frameworku Nette. Ačkoliv je tento framework za hranicemi České republiky a Slovenska praktický neznámý, umístil se na třetím místě a to díky tomu, že skupina českých respondentů, která jej jasně faforizovala, byla druhou nejpočetnější a pokud bychom přičetli ještě respondenty ze Slovenska, tvořila by tato česko-slovenská skupina vůbec nejpočetnější skupinu respondentů.

Pokud se však podíváme i na jiné zdroje, například na server Hongkiat.com (10 PHP Frameworks For Developers – Best Of, 2015), objevuje se většinou stále stejná skupinka několika frameworků, jako těch nejlepších.

Laravel

Laravel je mnohými považován za krále mezi frameworky. Tomuto tvrzení dává za pravdu i výše zmíněný průzkum popularity, kde Laravel naprosto dominoval. Ekosystém tohoto frameworku si získal svou popularitu velmi rychle a ve velkém měřítku a to i přesto, že byl vydán až v roce 2011 a jedná se o nejmladší z uvedených frameworků (History of Laravel PHP framework, 2013).

Od vydání se kolem něj utvořila obrovská podporující komunita a bylo vydáno mnoho různých tutoriálů a podkladů pro rychlé učení. Pro tyto účely vznikl dokonce speciální web Laracasts.com, který je plný různých videonávodů, které jsou řazeny dle úrovně složitosti od jednoduchých po ty složité. Díky tomu se mohou tento framework naučit používat i úplní začatečníci a to ve velmi krátkém čase.

Laravel je vytvořen tak, aby byl jednoduchý, snadno naučitelný a podporoval rapidní vývoj aplikací, což umožňuje vytvářet projekty jakýchkoliv velikostí v řádech dnů. Má také svůj vlastní odlehčený šablonový engine s názvem Blade.

Velkou výhodou Laravelu je také skutečnost, že zahrnuje lokální vývojové prostředí zvané Homestead, což je zabalený Vagrant box. To umožňuje začít vyvíjet

aplikace bez nutnosti instalovat webový server, HHVM, PHP či jakýkoliv jiný balíček na svůj lokální počítač nebo server. Mezi další důležité vlastnosti patří například:

- Podporuje autoloading, což je funkce díky které se automaticky načítají požadované třídy až v případě jejich potřeby.
- Snadné vytváření SQL dotazů pomocí nástroje, který umožňuje dotazy vytvářet prostřednictvím řetězení metod.
- Disponuje architekturou pro tvorbu REST API umožňující provádění CRUD (Create, Read, Update, Delete) operací pomocí HTTP požadavků (Laravel, 2016).

Symfony

Druhým nejpopulárnějším PHP frameworkem je Symfony. Tento framework je stejně jako Laravel dostupný pod licencí MIT, byl však vydán o dost dříve a to již v roce 2005 (Symfony Blog, 2007). Symfony je na svém oficiálním webu prezentován jako set znovupoužitelných komponent, které mohou vývojáři jednotlivě a nezávisle na frameworku využívat ve svých projektech. Tyto komponenty využívá v současné době mimo jiné také spousta zvučných projektů, mezi které patří například Drupal, phpBB, Joomla či Composer. Dokonce i již zmiňovaný Laravel využívá řadu knihoven Symfony (BrowserKit, Console, Debug, Routing a další).

Symfony se může pyšnit rozsáhlou celosvětovou komunitou, kterou si za léta své existence vybudoval. Samozřejmostí je podrobná dokumentace se spoustou dobře zpracovaných informací a tutoriálů, které jsou dostupné však pouze v textové podobě. Videa na způsob Laracastů zde budeme hledat marně.

Symfony2 používá šablonovací systém Twig, který umí dědění šablon, automatického escapování, filtry, funkce a další. Výhodou Twigu je především velmi dobrá podpora v IDE. Symfony2 totiž velmi dobře podporují populární vývojová prostředí jako NetBeans nebo PHPStorm (pomocí pluginů). Funguje zvýrazňování syntaxe šablonovacího systému Twig, našeptávání standardních anotací Symfony i Doctrine, našeptávání v konfiguračních souborech a mnoho dalšího (Zdroják, 2013).

Mezi další vlastnosti frameworku Symfony patří:

- Umožňuje aplikaci řídit příkazy příkazového řádku pomocí komponenty Console. Například pomocí příkazu cache:clear – vymažeme všechny keše. Lze vytvářet také vlastní příkazy.
- Snadná správa a instalace komponent pomocí Composeru.

Nette

Nette je český framework, který je napsaný v PHP 5 s plným využitím objektů (OOP) pro PHP. Ačkoliv vznikl už v roce 2004, zpřístupněn veřejnosti byl až v roce 2008. Původním autorem Nette Frameworku je David Grudl, o jeho další rozvoj se

stará organizace Nette Foundation (Nette Foundation, 2016). Nette Framework je šířen jako svobodný software, aby ho kdokoliv mohl používat a to i pro komerční účely. Můžete si vybrat, zda vám lépe vyhovuje licence New BSD nebo GNU General Public License (GPL) ve verzi 2 nebo 3.

Framework Nette čelí velké popularitě v domovské České republice a také na Slovensku, za hranicemi těchto států je však prakticky neznámý. O velké popularitě svědčí fakt, že disponuje nejaktivnější komunitou v České republice, která radí, vytváří rozšíření a sdílí své zkušenosti. Ve srovnání s jinými světově více známými frameworky však Nette v tomto ohledu výrazně zaostává. To stejné platí i o dokumentaci a dostupných tutoriálech, které nejsou tak kvalitní. Na Nette běží také mnoho významných českých webů jako například Česko-Slovenská filmová databáze ČSFD.cz nebo web jednoho z nejprodávanějších českých deníků Mladá fronta DNES a mnoho dalších.

Pokud se zaměříme na vlastnosti tohoto frameworku, lze zmínit především:

- Je založen na architektuře MVC.
- Nette používá revoluční technologii, která eliminuje výskyt bezpečnostních děr a jejich zneužití, jako je např. XSS, CSRF, session hijacking, session fixation atd.
- Disponuje ladicími nástroji, které vám pomohou zavčasu odhalit všechny chyby aplikace.
- Vyzrálý objektový návrh – promyšlený a čistý objektový návrh využívající nových vlastností PHP 5, komponent a událostmi řízeného modelování.
- Využívá moderní technologie jako jsou AJAX / AJAJ, Dependency Injection, SEO, DRY, KISS, MVC, Web 2.0, cool URL (Nette Foundation, 2016).

Yii2

Yii je velmi výkonný, komponentově založený PHP framework sloužící pro rapidní vývoj moderních webových aplikací. Název Yii (vyslovováno jako [ji:]) pochází z čínštiny a v překladu znamená „jednoduchý a vyvíjející se“. Tato zkratka je zároveň akronym pro anglické slovní spojení „Yes It Is“ (v češtině „Ano je!“), které má sloužit jako nejčastější odpověď na otázky jako: Je rychlý? Je bezpečný? Je vhodný pro můj projekt? Yes, it is!

Yii je framework, který je vhodný pro vytváření projektů libovolných velikostí založených na jazyku PHP. Díky své architektuře založené na komponentovém přístupu a sofistikované podpoře kešovaní, je vhodný zejména pro aplikace jako jsou nejrůznější portály, diskusní fóra, aplikace pro správu obsahu (CMS), e-shopy, RESTful webové služby a podobně.

Yii je v současné době dostupný ve dvou verzích, a to ve starší verzi 1.1, která je však stále podporována a ve verzi 2.0 což je kompletní přepsaná verze Yii přinášející podporu nejnovějších technologií a protokolů zahrnující Composer, PSR, jmenné

prostory atd. Verze 2.0 představuje současnou generaci frameworku a v následujících několik let bude věnováno vývoji právě této verze.

Klíčovými vlastnostmi Yii jsou:

- Jako většina PHP frameworků, Yii je založen na architektuře MVC a prosazuje organizaci kódu založenou na tomto vzoru.
- Fylosofie tohoto frameworku je, že kód by měl být zapsán jednoduchým a elegantním způsobem a striktně se zavazuje k dodržování návrhových vzorů.
- Yii je plnohodnotný framework poskytující mnoho prověřených a k okamžitému použití připravených funkcí.
- Je velmi dobře rozšiřitelný. Umožňuje upravit če zcela vyměnit téměř všechny své součásti.
- Je vysoce výkonný (The Definitive Guide to Yii, 2016).

Tento framework je také velmi přehledně a podrobně zdokumentován. Na oficiálních stránkách je dostatečné množství návodů a ukázkových kódů. Dokumentace je publikována rovněž v podobě tištěných knih. Kvůli rozsáhlému zájmu programátorů z různých zemí je stránka podpory rozdělena do různých jazykových kategorií.

CodeIgniter

Posledním z představených frameworků je CodeIgniter. Jedná se o open source framework, který je založený na architektonickém principu MVC a určený pro vývoj dynamických webových aplikací v PHP. Byl vyvíjen americkou společností EllisLab společně s rozsáhlou internetovou komunitou. První veřejná verze frameworku byla uvolněna v únoru roku 2006. V září 2014 společnost EllisLab oznámila, že vývoj bude pokračovat pod patronátem British Columbia Institute of Technology. V současné době je framework dostupný již ve své třetí verzi, která byla vydána v březnu roku 2015 a je dostupná pod MIT licencí.

Jádro tohoto frameworku je velice úsporně navrženo (velikost pouze zhruba 2MB a to včetně offline dokumentace). Lze jej však snadno rozšířit o velké množství různých doplňků. Díky tomu je celá struktura velmi přehledná. V důsledku jeho velké popularity a veřejného kódu může každý schopný programátor vytvořit nějaký doplněk. Díky tomu existuje mnoho nástrojů pto práci s e-maily, obrázky a hlavně dostupná komunikace s téměř každým používaným typem databáze. Rozdílem od jiných frameworků je, že nevyužívá šablony a tudíž není potřeba se učit nějaký jazyk určený právě k jejich tvorbě. CodeIgniter inponuje svojí uživatelskou základnou, jednoduchostí, dobrou dokumentací a uživatelským manuálem, který poskytne odpovědi prakticky na všechny řešené problémy.

Mezi další vlastnosti tohoto frameworku patří:

- Vysoký výkon.

- Nízka potřeba konfigurace.
- Velmi dobrá kompatibilita se všemi typy webových hostingů. CodeIgniter vyžaduje PHP ve verzi pouze 5.3.7 a podporuje většinu běžně používaných databázových systémů.
- Je založen na MVC, ale nevyžaduje ho. Povinný je pouze kontroler, kdežto modely a pohledy jsou nepovinné (CodeIgniter Web Framework, 2016).

2.4 Volba frameworku pro implementaci

Na základě výsledků provedé rešerše z oblasti PHP frameworků, byl vybrán framework Laravel jako ten nejvhodnější nástroj pro implementaci této práce.

Tento framework byl zvolen z více důvodů. Jedním z nich je jeho elegantní a intuitivní syntaxe, která se dá velmi rychle naučit. Laravel je také možné velmi snadno rozšiřovat pomocí různých balíčků, což je pro potřeby této práce důležité, jelikož bude řešena spousta problémů, kde vhodný balíček může být velmi nápomocný.

Dalším důvodem je jeho velká popularita, z které plynne řada dalších výhod. Laravel poskytuje na svých oficiálních stránkách výbornou dokumentaci a k dispozici je také server Laracasts.com, kde je možné shlédnout nejrůznější videonávody. V případě potřeby se lze obrátit také na některé ze známých internetových diskusních serverů. Zde si můžeme být, díky velkému počtu uživatelů Laravelu, prakticky jisti, že daný problém již někdo před námi řešil a že nalezneme odpověď.

3 Grafické uživatelské rozhraní webových aplikací

Grafické uživatelské rozhraní (Graphical User Interface – GUI) je kombinací technologií a prostředků, které umožnují uživateli komunikovat s počítačem a aplikací. Ve spojitosti s webovými aplikacemi se jedná především o grafické či textové prvky a jejich rozmístění. Tyto ovládací prvky umožňují uživateli práci s aplikací, získvají od uživatele potřebná vstupní data, reagují na výsledky zpracované aplikací a ty pak prezentují uživateli, v našem případě většinou prostřednictvím webového prohlížeče.

U moderních webových aplikací jsou při vývoji GUI aplikovány nejnovější technologie z této oblasti. Řeč je především o různých frameworcích, které nabízí širokou nabídku tzv. widgetů, které usnadňují a urychlují tvorbu designu aplikace. Dalším rysem moderních GUI je oddělení obsahu (výstupu aplikace) a grafické interpretace. Tento fakt má pozitivní vliv především při větších úpravách aplikace, aktualizacích, ale také při vytváření nových GUI aplikace pro jiné platformy (např. mobilní telefony). Aplikace tak může být z větší části zachována, není do ní třeba zasahovat a pouze se rozšíří o další GUI, což vede opět nejen k úspoře času, ale hlavně k úspoře ekonomických prostředků (Web design a uživatelská rozhraní, 2014).

3.1 Obecný postup při tvorbě GUI

Grafické rozhraní výrazně předurčuje to jak vnímáme celou aplikaci. Nejen že vytváří první dojem, ale také udává, jakým způsobem budou s aplikací uživatelé pracovat, zda s ní budou spokojeni a pomůže jim splnit jejich cíle. To vše má zásadní vliv na úspěch aplikace a proto nelze tuto součást vývoje podcenit a je třeba se jí patřičně věnovat.

V zásadě při tomto procesu usilujeme o dosažení co nejlepší použitelnosti aplikace. To zahrnuje dosažení co nejvyšší funkčnosti, efektivity, jednoduchosti, intuitivity a dobré zapamatovatelnosti. A to vše v pokud možno co nejlíbivějším designu. Obě dvě složky návrhu jsou stejně důležité a to jak použitelnost, tak estetická stránka. Hezky vypadající rozhraní, které není použitelné nebude plnit svůj účel tak jak má. To stejné bude platit pro aplikaci s výbornou použitelností avšak s nevzhledným designem, který zanechá v očích uživatelů špatný první dojem.

Obecně můžeme postup tvorby uživatelského rozhraní rozdělit do těchto vzájemně na sobě navazujících etap:

1. Vytvoření strategie
2. Specifikace požadavků
3. Návrh struktury aplikace
4. Vytvoření základního modelu a kostry
5. Návrh designu aplikace
6. Testování

Vytvoření strategie

Prvotním krokem při procesu tvorby uživatelského rozhraní je vytvoření strategie. Tento krok tvoří základ pro všechny následující. Úkolem vytvoření strategie je najít odpovědi na základní avšak velmi podstatné otázky jako jsou: Jaký je účel aplikace? Co aplikace přinese provozovateli? Co uživatelé od aplikace očekávají a co jim přinese? Jaká je cílová skupina uživatelů?

Pochopení cílové skupiny uživatelů a odhalení jejich skutečných potřeb je velmi důležité. Pokud se nám podaří tyto informace správně odhalit, můžeme pak na základě těchto poznatků navrhnout GUI s ohledem přesně na tyto požadavky a cíle provozovatele aplikace. Pro určení potřeb uživatele je využívána celá škála nástrojů, od segmentace uživatelů aplikace a jejich charakteristiky, přes skupinové diskuze a interview, po vytváření scénářů a profilů typických uživatelů – tzv. person, což jsou fiktivní profily postav, které reprezentují určitou skupinu svojí charakteristikou nebo specifickým chováním.

Specifikace požadavků

Úkolem této etapy je na základě již získaných poznatků specifikovat funkce a možnosti, které by měla aplikace poskytovat. To vše zatím v obecné rovině. Neřeší se tedy konkrétní implementace funkcí, ale to jaké funkce bude GUI poskytovat. Stejně tak by měly být specifikovány požadavky uživatelů na formu a obsah rozhraní. Veškeré tyto poznatky by měly být zdokumentovány, což později velmi usnadní práci grafickým designerům při návrhu designu jednotlivých prvků aplikace. Takto vytvořená specifikace hraje důležitou roli také při volbě vhodných technologií pro implementaci.

Strategická část spolu se specifikací požadavků a obsahu návrhu také bývá označována jako analýza potřeb, zahrnující jak uživatele, tak provozovatele aplikace. Dobrá znalost požadavků velmi ulehčí vývoj struktury celé aplikace. Výsledkem specifikační části je dokument s popisem požadovaných funkcí a formy obsahu v souvislosti s definovanou strategií.

Návrh struktury aplikace

Tato fáze vývoje aplikace a jejího GUI přebírá výsledky předchozích analýz a dává do spojitosti jednotlivé funkce a formu obsahu. Zabývá se především interakcí mezi jednotlivými definovanými funkcemi (interakčním designem), tedy jak budou budoucí uživatelé procházet jednotlivými kroky ke splnění určitého dílčího úkolu a jak bude rozčleněn obsah aplikace s ohledem na vzájemné vazby.

Pro tuto fázi vývoje je charakteristické vytváření velkého množství diagramů, které zachycují jednotlivé vazby mezi funkcemi nebo kategoriemi obsahu. Tyto vazby jsou opět velmi důležité pro samotné návrháře grafického rozhraní, neboť lze podle nich určit nejvhodnější rozmístění komponent či výběr konkrétního GUI prvku pro daný úkol a funkcionalitu.

Výsledkem je množství tzv. Use Case diagramů, které reprezentují nejčastější scénáře, případy užití, vazby mezi jednotlivými částmi aplikace a také akce, které bude uživatel moci provádět, včetně možných reakcí aplikace. Tyto schematické diagramy jsou pak využity v další fázi při návrhu již konkrétního rozvržení základních oblastí uživatelského rozhraní.

Vytvoření základního modelu a kostry

Poté co je definován obsah a souslednost jednotlivých aplikací, přichází na řadu tvorba grafického modelu a kostry grafického uživatelského rozhraní. V této fázi vývoje jsou propojeny dohromady tři různé části návrhu. První z nich je informační design, který klade důraz na prezentaci informací a to tak, aby došlo k jejich správnému pochopení. Další složkou je tzv. interface design, který řeší otázky výběru jednotlivých druhů GUI komponent k provedení určité činnosti. Třetí složkou je design navigace, jež má za úkol vyřešit, jak se budou v prostředí aplikace její uživatelé pohybovat.

Celkový pohled lze pak získat sdružením těchto poznatků a vytvořením tzv. drátěného modelu uživatelského rozhraní – wireframe. Wireframe definuje rozmístění funkčních prvků na stránce. Nejedná se v žádném případě o grafický návrh, neobsahuje obrázky a je tvořen pouze pomocí čar a textu. Nedoporučuje se ani použití barev, až na výjimky, které je potřeba odlišit. K tvorbě wireframů lze použít speciální software, ale lze si vystačit i s papírem a tužkou.

Největší výhodou takovýchto náčrtků je jejich flexibilita, možnost je rychle měnit a vytvářet více variant řešení. Lze také snáze testovat, jak se s rozložením prvků stránky popasují sami uživatelé. Na základě uživatelského testování pomocí wireframů lze vybrat nejlepší rozložení GUI prvků, což omezuje selhání a chyby v navrhování uživatelského rozhraní. Podle jednotlivých wireframů mají designeři GUI komponent lepší představu o jejich funkci, umístění, přibližné velikosti a celkovém začlenění v aplikaci.

Návrh designu aplikace

Tato část návrhu GUI je ta, která je označována jako webdesign. Hlavní část práce je nyní na grafických designerech, kteří připravují vizuální vzhled jednotlivých komponent stránky, podle vytvořených wireframů a storyboardů. To zahrnuje vytvoření barevných schémat a výběr fontů, které budou v aplikaci použity a vytvoření jednotného vzhledu všech použitých komponent. Stejně jako u tvorby wireframů je i zde průběh laděn k všeobecné spokojenosti opakujícími se cykly vývoj – prezentace – implementace změn.

Testování

Poslední fází vývoje je testovaní již hotové aplikace. Testováním se získává zpětná vazba od uživatelů a jsou odhalovány nedostatky a příležitosti pro vylepšení. Tyto poznatky jsou promítány buď do stádia wireframů nebo dílčích změn v designu

GUI. Uživatelské testování může probíhat jak v uzavřené skupině, tak v testovacím provozu aplikace za pomoci speciálních analytických nástrojů pro web (analýzy návštěvnosti, teplotní mapy, sledování pohybů myši a kliknutí, nahrávání uživatelské aktivity a mnohé další).

Změny a úpravy jsou při testování přirozené, nemělo by však v kvalitně zpracovaném projektu dojít k situaci, kdy je potřeba předělat velkou část aplikace. To ukazuje na zanedbání některých fází vývoje (Web design a uživatelská rozhraní, 2014).

3.2 Technologie pro tvorbu GUI

Grafické uživatelské rozhraní webových aplikací je realizováno kombinací jazyků HTML, CSS a JavaScript. HTML představuje strukturu GUI, tedy jaké prvky jsou obsaženy a jaká je mezi nimi hierarchie. To jak jednotlivé prvky vizuálně vypadají má na starost CSS. Pomocí JavaScriptu je pak realizována interaktivita a to tak, že jsou prvkům přiřazeny reakce na události, jako například kliknutí myši nebo přejetí kursorem myši.

Dobrým zvykem, který výrazně přidává na přehlednosti a usnadňuje práci v případě jakýchkoliv úprav a zásahů do aplikace, je nemíchat všechny tyto technologie do jednoho souboru, ale uchovávat je odděleně ve speciálních souborech – tedy zvlášť HTML, zvlášť CSS a stejně tak i JavaScript. V takovém případě pak můžeme například zcela změnit vzhled aplikace pouhým nahrazením souboru s CSS styly. Další možnost jak si ulehčit práci přináší použití některého z frameworků.

3.3 Nástroje používané při tvorbě GUI webových aplikací

Pro tvorbu GUI v dnešní době existuje celá řada knihoven a frameworků. V případě frameworků pro tvorbu GUI hovoříme o tzv. front-end frameworcích. Tyto frameworky jsou skvělým nástrojem, který značně urychluje vývoj GUI webových aplikací. Představují sadu předem připravených prvků (tlačítka, formuláře, tabulky, navigace apod.), které jsou připraveny k okamžitému použití. Navíc poskytují bonusy v podobě kompatibility napříč webovými prohlížeči nebo responzivity a další. Vývojaři tak odpadá nutnost optimalizace a testování kódu pro různé typy pohlížečů a zařízení, ale může se soustředit na hlavní část vývoje.

Použitím front-end frameworku tak vývojář nemusí s každým novým projektem začínat od nuly. Díky tomu jsou vynikájícím nástrojem také pro vytváření různých prototypů, kdy není až tak důležitý konečný vzhled GUI, ale rychlost vytvoření. Podobných frameworků je dostupných celá řada, za zmínku však stojí především dva z nich a to frameworky Bootstrap a Foundation.

Bootstrap

Nejvíce oblíbeným mezi front-end frameworky je Bootstrap. Bootstrap vyvinuli Mark Otto a Jacob Thornton, kteří té doby pracovali pro společnost Twitter Inc. Původně se mělo jednat o interní nástroj společnosti, avšak tito vývojáři viděli příležitost udělat něco více a tak vznikl Bootstrap, který byl v srpnu roku 2011 vydán jako open source (Mark Otto, 2012).

Bootstrap je volně stažitelná sada nástrojů pro tvorbu webu a webových aplikací. Svou polularitu si získal především díky své jednoduchosti použití. Díky tomu jej mohou využívat uživatelé všech úrovní znalostí. Framework nabízí širokou škálu všech běžně používaných komponent v oblasti GUI webových aplikací.

Framework si zakládá na heslu „Jeden framework, veškerá zařízení“. To znamená že se efektivně stará o to, aby se aplikace zobrazila vhodným způsobem na jakémkoliv typu zařízení (od mobilů až po velké monitory desktopů). K tomuto účelu využívá CSS media Queries a na nich postavený tzv. mřížkový systém (Grid system). Ten je založen na principu řádků a sloupců, kde v jednom řádku může být až 12 sloupců.

Foundation

Druhým velmi populárním front-end frameworkem je Foundation. Foundation je open source projekt firmy ZURB vyvinutý za účelem urychlení a zlepšení vývoje front-endu u webových aplikací. První verze byla vydána v roce 2011. V současné době je dostupná již šestá verze (Foundation, 2016).

Tento framework není až tak populární jako Bootstrap, ale svými kvalitami se mu vyrovná a v některých aspektech jej i předčí. Foundation nabízí podobnou sadu prvků jako Bootstrap. Ve svém základu má však také některé zajimavé prvky, které u Bootstrapu nenajdeme – například prvky jako Slider, Clearing Lightbox nebo Joyride. Také Foundation je framework pro tvorbu responsivních webů založených na systému mřížky.

jQuery

Jak už bylo zmíněno, GUI webových aplikací je výsledkem kombinace CSS, HTML a JavaScriptu. Front-end frameworky usnadňují práci tím, že poskytují sadu předpřipravených prvků a díky tomu vývojáři nemusí trávit tolik času sestavováním těchto prvků v HTML a jejich stylizací pomocí CSS. Co tyto frameworky však za vývojáře neudělají, je vytvoření jejích interakcí (například co se stane, když uživatel klikne na dané tlačítko).

O interakce prvků GUI se v moderních webových aplikacích stará JavaScript většinou ve spojení s technologií AJAX (Asynchronous JavaScript and XML), díky které můžeme měnit obsah webových stránek bez nutnosti jejich kompletního znovunačítání. Aby tedy dané prvky GUI poskytovali požadovanou funkcionalitu, je potřeba aby je vývojář správně naprogramoval právě prostřednictvím JavaScrip-

tu. Zde se nabízí opět dvě možnosti. Tou první je psát tento kód pomocí čistého JavaScriptu. Druhou možnost pak představuje sáhnutí po některé z knihoven.

Knihoven pro JavaScript existuje celá řada – například jQuery, React nebo Angular. Zde bude představena pouze jedna z nich a to jQuery. Tato knihovna je v současné době nejčastěji používanou JavaScriptovou knihovnou. Také oba představené front-end frameworky využívají jQuery pro některé základní interakce svých prvků (například zobrazení modálního okna po kliknutí na tlačítko).

JQuery se vyznačuje především velmi jednoduchou syntaxí. Další velkou výhodou je fakt, že si knihovna sama řeší nekompatibilitu mezi prohlížeči. Není tedy potřeba se tímto problémem zabývat a můžeme se spolehnout na to, že kód bude fungovat správně ve všech prohlížečích. Kromě toho velmi ulehčuje manipulaci s obsahem webu. Používá tzv. CSS selectory, pomocí kterých je možné velmi snadno pracovat s jednotlivými prvky.

Příklad použití jQuery demonstruje následující jednoduchá ukázka, kde se nachází jedno tlačítko:

```
1 | Click Me!
```

Uvedenému tlačítku je přiřazen identifikátor `alert_btn`, za pomoci kterého je možné toto tlačítko identifikovat a naprogramovat reakce na události, které nastanou právě při interakci s tímto tlačítkem. Příkladem může být přiřařazení reakce na událost kliknutí tlačítka myši:

```
1 | $( "#alert_btn" ).click(function(){
2 |   alert( "Clicked!" );
3 | });
```

Tlačítku je nyní přiřazena reakce na událost click v podobě funkce, která zobrazí hlášku (alert box) s textem „*Clicked!*“. Tato funkce bude vyvolána v případě, že dojde ke kliknutí myší na dané tlačítko.

4 Analýza zadání od zákazníka

Abychom mohli co nejlépe porozumět zadání od zákazníka, je vhodné, se na počátku všeho nejprve seznámit s firmou samotnou, zjistit jak firma funguje, jakou provozuje činnost a jaké nabízí produkty či služby.V této kapitole je nejprve krátce představena firma Metal Ryspol, s. r. o. a její činnost. Dále se text zabývá současným stavem webových stránek této firmy a problémy s nimi spojenými. V neposlední řadě je provedena analýza zadání od firmy.

4.1 Představení firmy Metal Ryspol, s. r. o.

Metal Ryspol, s. r. o. je malou firmou, která je tvořena čtyřmi spolumajiteli a jedním zaměstnancem. Firma sídlí v obci Tvrdonice, která se nachází nedaleko města Břeclav a na trhu působí od roku 1994. Předmětem její činnosti je zakázková kovovýroba. Pod tímto pojmem si můžeme představit všemožné kovové výrobky a to od těch nejmenších, jako jsou například drobné soustružnické práce, až po konstrukce menších či středně velkých hal.

Firma za dobu své existence prošla několika změnami – jednak změnou jména a stěhovaním do větších výrobních prostor v roce 2003, měnilo se však také primární zaměření firmy a to jaké výrobky a služby firma poskytovala. V současnoti se zaměřuje především na výrobu boxů pro koně. To je zapříčiněno tím, že chov koní si v České republice získává v posledních letech stále větší a větší oblibu. Pokud se podíváme na statistiky, tak v roce 2003 bylo v česku chováno asi 41 000 koní, od té doby se tento počet více než zdvojnásobil a v současnosti je v naší zemi chováno asi 85 000 koní (Novinky, 2016).

S růstem počtu koní roste také poptávka po ustájení koní. Proto stále více lidí má zájem o výrobu boxů pro koně a to buď pro potřeby svých vlastních koní nebo za účelem dalšího pronájmu boxů.

Boxy pro koně

Pod pojmem boxy pro koně si lidé, kteří se o chov koní příliš nezajímají často mylně představují úplně jiné věci než jimi opravdu jsou – jako například různé přepravníky pro koně nebo přívěsy za auto. Proto je pro úplnost vhodné zmínit co to ty boxy opravdu jsou.

Pojem box pro koně představuje prostor ve stáji, kde je kůň ustájen. Typicky je v jedné stáji ustájeno vícero koní, kde každý kůň má svůj vlastní prostor, který je vymezen pomocí různých dělících stěn a přepažek. Právě tento vymezený prostor je pak označován za box pro koně. Pokud tedy chovatel koní poptává boxy pro koně, pak hledá někoho, kdo mu vyrobí tyto dělící stěny.

Boxy pro koně se pak dělí na vnitřní a venkovní. V případě vnitřních boxů zákazník vlastní nějakou budovu a poptává pouze vnitřní stěny do této budovy. V případě venkovních boxů je zahrnuto také postavení budovy či přístřešku. V takovém případě se jedná o projekt, který vzniká tzv. „na zelené louce“.

Obrázek 3: Ukázka typických boxů

Typický box pro koně je tvořen jednou přední stěnou s dveřmi, jednou či dvěma bočními stěnami, které slouží pro oddělemí jednotlivých boxů od sebe a také buď dalšími dveřmi či oknem na zadní stěně. V případě, že box má na zadní stěně dveře, je možné k němu volitelně přidat také tzv. paddock, což je venkovní výběh pro koně. Na obrázku číslo 3 jsou zobrazeny ukázky typických boxů pro koně – vlevo vnitřní a vpravo venkovní.

4.2 Současné webové stránky

Existence webových stránek je pro chod firmy zásadní. Prakticky veškeré zakázky jsou získávány právě prostřednictvím webu. Pokud se podíváme blíže na současné webové stránky firmy, hned na první pohled jsou zde patrné nedostatky. Z nichž hlavním je neaktuálnost informací.

Dále je patrné, že obsah webu je již roky nezměněn, například v sekci fotogalerie nalezneme poslední záznam z roku 2009, což může vyvolávat dojem že firma je tzv. „mrtvá“ a již nefunguje. To je způsobeno tím, že web je statický – není napojen na žádný redakční systém a ve firmě není nikdo kdo by uměl takový web spravovat.

Další problém tkví v tom, že obsah webu není v souladu se strategií firmy. Tím je na mysli fakt, že zde nalezneme spoustu sekcí, které nejsou podstatné a navíc způsobují to, že hlavní a důležité informace zaniknou. Mnoho návštěvníků, kteří přijdou na web pomocí internetového vyhlédávače za účelem boxů pro koně, pak rychle tento shluk různých informací odradí a vyvolá dojem, že jsou na špatné stránce. V takovém případě naštěvník web rychle opouští a firma přichází o potenciálního zákazníka. V neposlední řadě design webu je rovněž zastaralý a není optimalizovaný pro mobilní zařízení.

Obrázek 4: Současné webové stránky firmy Metal Ryspol, s. r. o.

4.3 Specifikace požadavků

Specifikace požadavků představuje první krok při vývoji jakéhokoliv softwaru či aplikace a je potřeba se jí věnovat velmi důkladně. Správně specifikované požadavky mají zásadní vliv na úspěšné dokončení vývoje projektu. V opačném případě, kdy na začátku nebyly správně pochopeny opravdové potřeby zadavatele, dochází k situacím, kdy je implementováno něco co vůbec nebylo požadováno nebo způsobem, který neodpovídá představám zadavatele.

Požadavky lze rozdělit do dvou skupin a to na funkční a nefunkční. V případě funkčních požadavků se jedná o požadavky na konkrétní funkce, které by měla aplikace nebo systém vykonávat. Nefunkční požadavky pak představují vlastnosti systému jako celku a definují také různá omezení.

Funkční požadavky

Jak už bylo uvedeno dříve, firma je v situaci, kdy disponuje zastaralým webem, který zcela nedostačuje jejím potřebám. Proto požaduje vytvoření nového firemního webu včetně administrace, pomocí které by byl web spravován. Konkrétní funkční

požadavky jsou uvedeny zvlášť pro veřejnou část a zvlášť pro administrační část. V případě veřejné části systému evidujeme tyto požadavky:

- Na úvodní stránce se budou prostřednictvím slideru zobrazovat názory zákazníků.
- Na webu bude zobrazen seznam produktů firmy. V detailu každého z produktů se bude zobrazovat video, popis a fotogalerie produktu.
- Součástí webu bude fotogalerie. U každého obrázku bude zobrazen popisek.
- Na webu se budou zobrazovat reference pomocí bodů na mapě, kde se po kliknutí na konkrétní bod zobrazí detail reference.
- Uživatelé budou moci na webu zažádat o vytvoření kalkulace pomocí formuláře.
- Součástí bude také konfigurátor, který umožní uživatelům vytvářet 3D modely venkovních boxů pro koně. Konfigurátor bude schopen měnit parametry venkovních boxů, dále použité materiály a také barvy. Tyto modely bude možné ukládat nebo zasílat ve formě žádosti o kalkulaci. Uživatelé si budou moci své uložené modely nahrát a následně opět upravovat.
- Web bude také poskytovat návštěvníkům možnost kontaktovat firmu pomocí formuláře.

K administrační části byly shromážděny tyto funkční požadavky:

- **Přehled** – Systém by měl mít stránku s aktuálním přehledem, kde budou zobrazovány informace o aktuálních zakázkách a nových žádostech o kalkulaci.
- **Správa uživatelů** – Přístup do systému bude na základě uživatelského jména a hesla. Systém bude evidovat dva typy uživatelů – administrátora a běžného uživatele. Administrátor bude mít právo vytvářet, upravovat a mazat uživatele systému.
- **Správa textového obsahu** – Systém bude umožňovat snadno spravovat obsahu webu. Bude možné vytvářet nové dokumenty, upravovat stávající a mazat je.
- **Správa fotogalerií** – Pomocí systému bude možné snadno nahrávat obrázky do fotogalerií a to bez nutnosti jakýchkoliv úprav. Systém by měl umět automaticky vytvářet náhledy obrázků. Jednotlivé obrázky bude možné mazat, skrývat a upravovat jejich popisek.
- **Správa reference** – Systém bude poskytovat nástroje pro správu referencí. Jednotlivé reference by mělo být možné vytvářet, upravovat, skrývat a mazat.
- **Správa názorů zákazníků** – Pomocí systému bude možné přidávat, upravovat a mazat názory zákazníků, které se budou zobrazovat na domovské stránce webu.

- **Správa produktů** – Bude možné vytvářet, upravovat, skrývat a mazat jednotlivé produkty.
- **Správa konfigurátoru boxů** – Systém bude spravovat materiály a barvy, které budou zobrazeny v konfigurátoru boxů. U každého materiálu bude možné definovat jeho vlastnosti a přiřadit texturu, miniaturu a zvolit jeho barevné varianty. Veškeré materiály a barvy bude možné vytvářet, upravovat, skrývat a mazat.
- **Správa kalkulací** – V systému bude možné spravovat žádosti o kalkulaci. Součástí bude také nástroj pro výpočet celkové výše kalkulace, která bude stanovena na základě vstupních informací o boxech (počet, typ, materiály, barvy). Vypočtené kalkulace bude možné odeslat zákazníkovi formou e-mailu.
- **Správa zakázek** – Systém bude evidovat zakázky. Ke každé zakázce bude možné přiřadit termín dodání. Systém by měl umět také rozlišovat aktuální zakázky a zakázky, které již byly zkompletovány.

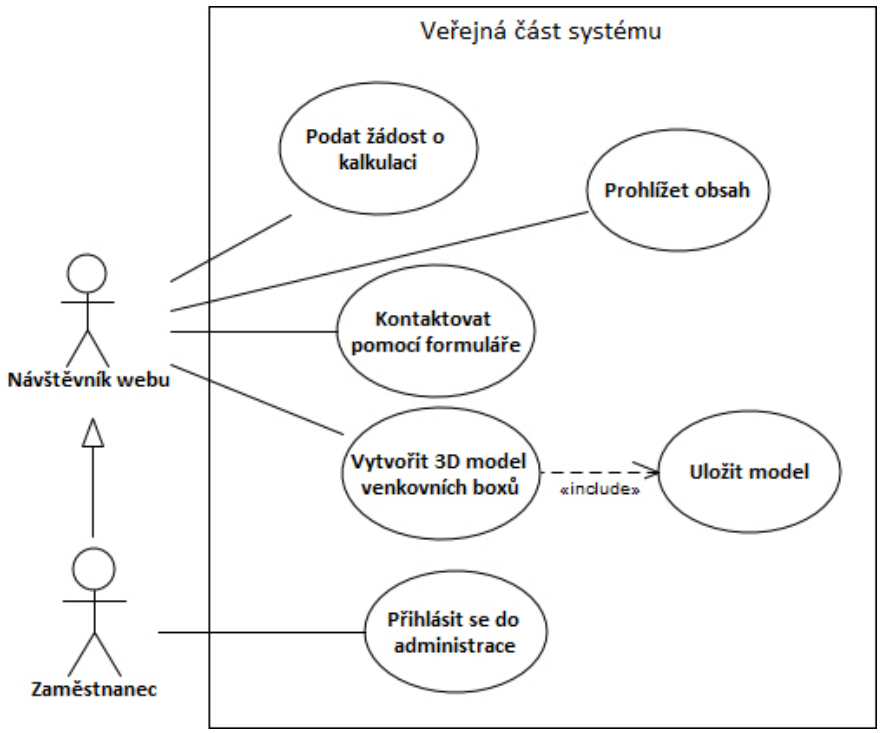
Nefunkční požadavky

Nefunkční požadavky je potřeba neustále promítat do návrhu řešení, jelikož nedodržení některého z nich může mít zásadní vliv na aplikaci a způsobit velké komplikace. V tomto případě byly identifikovány tyto nefunkční požadavky:

- Požadavek na responzivní design.
- Systém by měl fungovat na současném webhostingu.
- Veřejná část by měla mít moderní design a přehledné uspořádání.
- Vše by mělo být pokud možno intuitivní a snadno použitelné i pro laika.

4.4 Diagram případů užití pro veřejnou část systému

Use Case Diagram (česky diagram případů užití) zobrazuje chování systému tak, jak ho vidí uživatel. Účelem diagramu je popsat funkcionalitu systému, tedy co od něj klient nebo my očekáváme. Diagram vypovídá o tom, co má systém umět, ale neříká jak to bude dělat. Use Case diagram se skládá z případů užití (use case), dále aktérů (actors) a vztahů mezi nimi (ITnetwork, 2013).

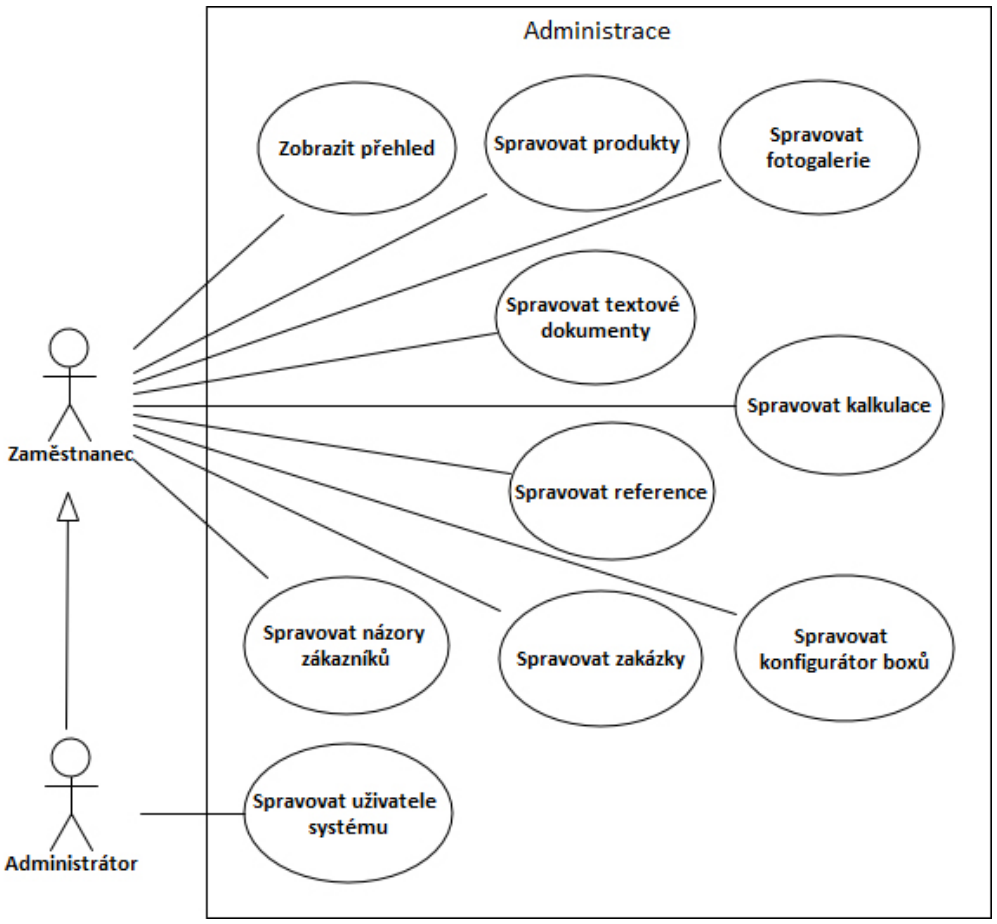


Obrázek 5: Diagram případů užití pro veřejnou část systému

Z obrázku č. 5, který znázorňuje diagram případů užití pro veřejnou část systému, můžeme vypozorovat, že zde vystupují 2 typy aktérů. Jedná se o návštěvníky webu, kteří mohou jednak prohlížet obsah webu, ale také vytvářet žádosti o kalkulaci, kontaktovat firmu pomocí formuláře nebo vytvářet 3D modely venkovních boxů pro koně. Druhý aktér – zaměstnanec, pak může provádět stejné akce jako návštěvník webu (tento vztah je znázorněn formou dědičnosti), navíc má možnost přihlásit se do administrace.

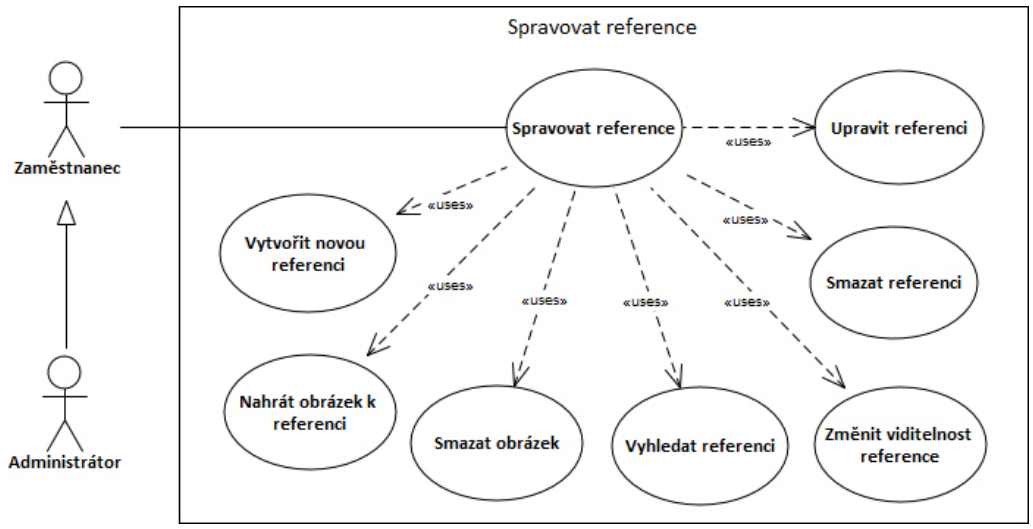
4.5 Diagram případů užití administrační části systému

Use Case Diagram pro administrační část systému je již o poznání komplikovanější. Vzhledem k tomu, že administrace není určena pro běžné návštěvníky, vystupují zde v podobě aktérů pouze zaměstnanci respektive administrátoři. Administrátor je zaměstnanec, který má navíc právo spravovat uživatele systému tzn. vytvářet, měnit nebo upravovat uživatelské účty. Následující diagram popisuje případy užití administrační části z nejvíce obecného pohledu.



Obrázek 6: Diagram případů užití pro veřejnou část systému

Pokud si jednotlivé případy užití z výše uvedeného diagramu rozebereme s větší podrobností, zjistíme že za každým z nich se skrývá řada dalších dílčích případů užití. Ve většině z nich se jedná o případy užití jako vyhledání záznamu, vytvoření nového záznamu, upravení již existujícího, nastavení viditelnosti, smazání záznamu či vytvoření vazby mezi záznamy. Pokud se tedy zaměříme například na správu referencí, dostaneme následující diagram:



Obrázek 7: Diagram případů užití pro správu referencí

4.6 Specifikace případů užití

V jednotlivých diagramech případů užití je naznačeno, jak aktor používá systém. Jsou zde však uvedeny pouze názvy činností, které nejsou dále nijak specifikovány, což není pro účely analýzy požadavků dostatečné. Případy užití je potřeba doplnit o textovou specifikaci, která jednoznačně popisuje činnost, kterou daný případ užití vykonává.

Specifikace nemá žádnou pevně definovanou podobu, může být například ve formě tabulky nebo prostého textu. Obsahuje jednotlivé případy užití a ke každému z nich definuje několik bodů:

1. **Krátký popis** – Měl by vysvětlovat, jakou má funkčnost pro uživatele přidanou hodnotu, proč ji uživatel spouští.
2. **Aktéři** – Tato část jmenuje aktéry, kteří se případu užití účastní.
3. **Podmínky pro spuštění** – Každý případ užití může mít definované určité podmínky, které musí být pro jeho spuštění splněny.
4. **Základní tok** – V jeho bodech je popsána interakce mezi aktéry a jednotlivými případy užití. Body zapisujeme jako scénář, ve kterém se střídají vždy aktér a systém.
5. **Alternativní toky** – Specifikace může obsahovat několik alternativních toků (scénářů), které umožňují reagovat na odchylky od scénáře hlavního.
6. **Podmínky pro dokončení** – Podobně, jako může mít případ užití podmínky přes spuštěním, může mít i podmínky pro dokončení (ITnetwork, 2013).

Specifikace případu užití – Vytvořit novou referenci

Konkrétní příklad specifikace případu užití je znázorněn na případu vytvoření nové reference. Budeme-li postupovat podle výše uvedených bodů, pak tato specifikace může mít následující podobu:

Tabulka 1: Specifikace případu užití – Vytvořit novou referenci

Případ užití	Vytvořit novou referenci
Popis	Případ užití umožňuje vytvořit novou referenci.
Aktéři	Zaměstnanec (Administrátor), Systém
Podmínky pro spuštění	Zaměstnanec musí být přihlášen do administrace.
Základní tok	1. Zaměstnanec aktivuje případ užití kliknutím na tlačítko „Přidat novou referenci“. 2. Systém zobrazí formulář pro zadání vstupních dat. 3. Zaměstnanec vyplní zobrazený formulář a odešle jej. 4. Systém zvaliduje data od zaměstnance. 5. Systém uloží referenci a informuje uživatele o úspěšném provedení operace.
Alternativní tok	3.1 Pokud zaměstnanec zadal neplatný vstup nebo vstupy, systém na skutečnost uživatele upozorní a nedovolí referenci vytvořit. 3.2 Zaměstnanec opraví neplatný vstup či vstupy a tok pokračuje na 3. bodu základního toku.
Podmínky pro dokončení	Nová reference bude korektně uložena v databázi.

4.7 Diagramy aktivit

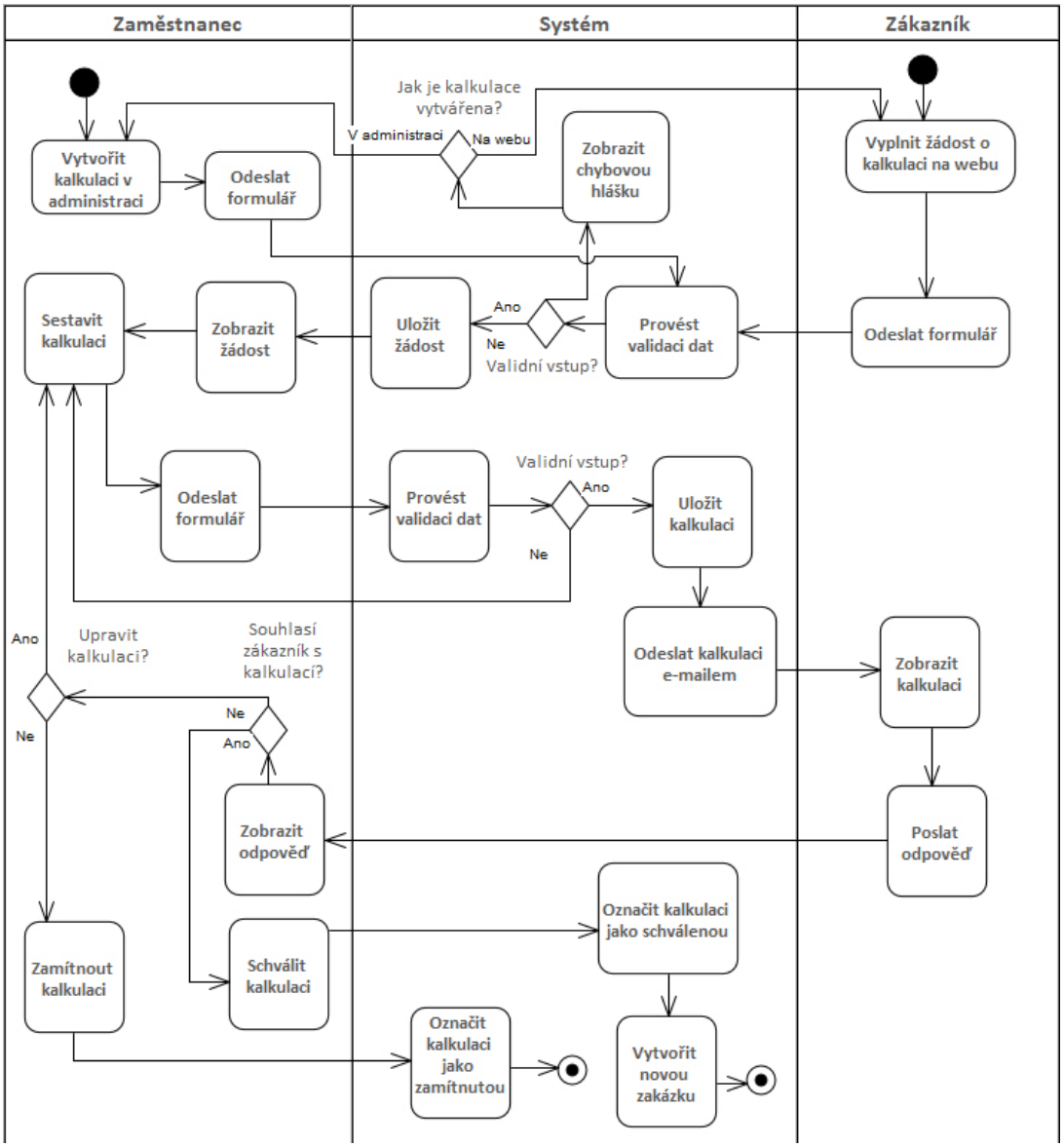
Jednotlivé případy užití či procesy je vhodné v některých případech specifikovat také pomocí grafické formy. Jedná se o situace, kdy je například daný proces důležitý nebo pokud není úplně triviální. K tomuto účelu je možné použít tzv. diagramy aktivit.

Diagram aktivit je typem diagramu interakcí, který se používá pro popis procedurální logiky, byznys procesů či pracovních postupů. Umožňuje také graficky modelovat jednotlivé případy užití jako posloupnost akcí.

Diagram aktivit modeluje procesy jako aktivity, které se skládají z uzlů vzájemně propojených hranami. Existují tři typy uzlů – akční uzly, které reprezentují samostatné a v rámci aktivity nedělitelné jednotky, řídící uzly, jejichž úkolem je řídit cestu uvnitř aktivity a uzly objektové, které zastupují objekty. Nejpoužívanějším akčním uzlem je tzv. call action node, který inicializuje aktivitu, chování či operaci. Příkladem řídících uzlů jsou počáteční, konečné uzly nebo uzly rozhodnutí. Pro

zpřehlednění se může diagram rozdělit například dle rolí či organizačních jednotek do tzv. zón odpovědnosti či plaveckých drah. (ITnetwork, 2013).

Diagram aktivit pro proces vytvoření nové zakázky



Obrázek 8: Doménový model

K tomu aby mohla být vytvořena nová zakázka, je nutné, aby nejprve byla vytvořena kalkulace. Kalkulace může vzniknout dvěma způsoby. Jednak si o její vytvoření může zažádat zákazník prostřednictvím formuláře, který je dostupný na webu. Druhou možností je pak vytvoření kalkulace zaměstnancem v administraci. V obou přípa-

dech systém kontroluje správnost zadávaných vstupních údajů a v případě chyby na tuto skutečnost upozorní. Je-li všechno v pořádku, systém novou žádost uloží do databáze.

Nová žádost je dále v režii zaměstnance, který si ji zobrazí a sestaví cenovou nabídku pro zákazníka. Kalkulace je poté odeslána a zpracována systémem, který i zde provede validaci a v případě úspěchu spočítanou kalkulaci uloží a odešle ji zákazníkovi ve formě e-mailu. Poté co se zákazník vyjádří k nabídce, provede zaměstnanec operaci odpovídající této odpovědi. V případě, že odpověď je negativní, má zaměstnanec dvě možnosti, buď může kalkulaci označit za zamítnutou a tím celý proces skončí nebo může sestavit vylepšenou kalkulaci a tu opět odeslat zákazníkovi. Je-li však odpověď zákazníka na cenovou nabídku pozitivní, zaměstnanec zadá do systému požadavek o schválení kalkulace, systém tento požadavek zpracuje a vytvoří novou zakázku.

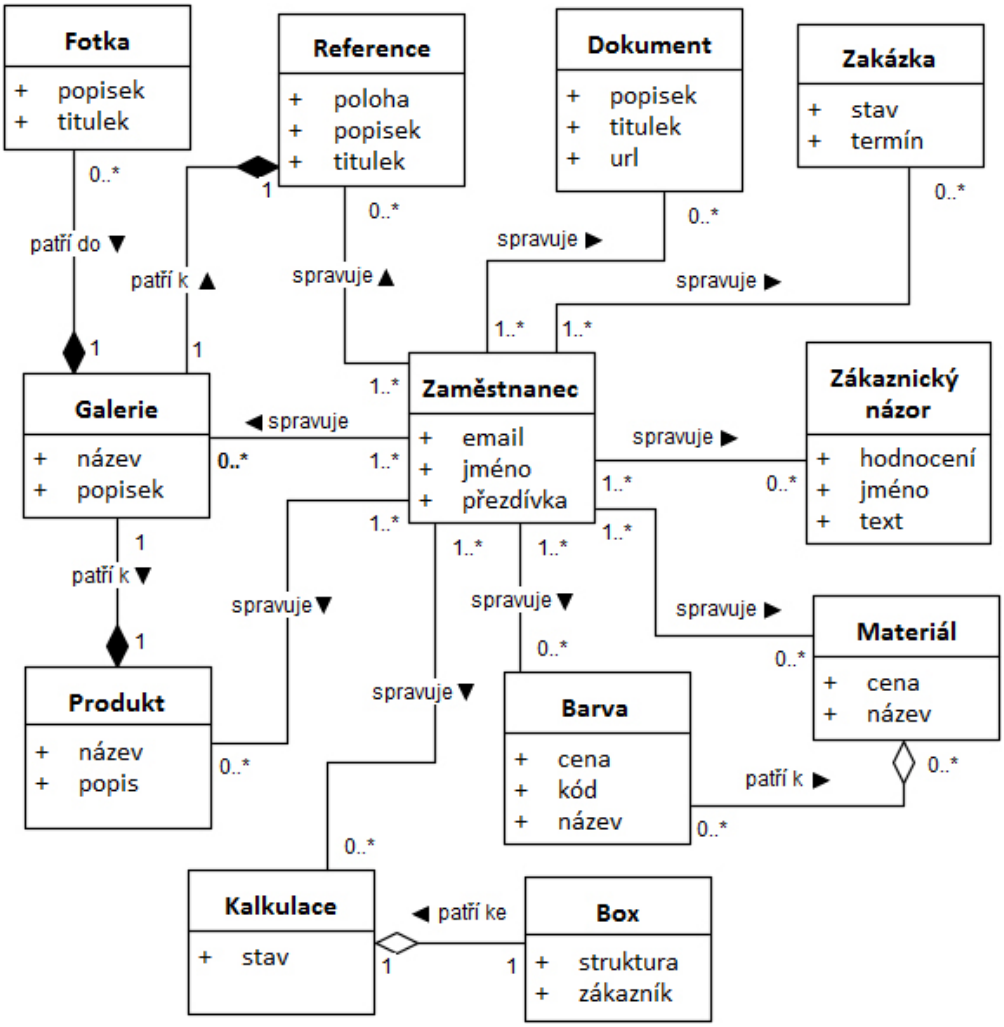
4.8 Doménový model

Poté co je vytvořena specifikace případů užití, můžeme přistoupit k modelování tzv. doménového modelu. Jedná se o formu class diagramu, tedy diagramu tříd. Třídy v doménovém modelu jsou značně zjednodušené, neobsahují metody a mají pouze důležité atributy. Názvy tříd, atributů a další identifikátory můžeme psát s diakritikou. Model je tedy jakýsi náčrt základních entit systému a vztahů mezi nimi. Je platformově nezávislý (není určen pro konkrétní programovací jazyk) a atributy nemají datové typy.

Při tvorbě doménového modelu vycházíme ze zadání klienta. Z něj identifikujeme klíčové entity a vztahy mezi nimi. Tyto entity zakreslíme do modelu jako třídy. Grafická notace třídy je obdélník rozdělený vodorovně na 3 části. V první je zapsané jméno třídy, v druhé jsou její atributy a ve třetí části nalezneme metody. Pro doménový model uvádíme jen zjednodušenou notaci s názvem třídy a atributy.

Třídy mohou být mezi sebou propojeny pomocí vztahů:

- **Asociace** – Asociace určuje základní vztah mezi dvěma entitami. Ty mohou existovat nezávisle na sobě. Zakreslujeme ji jako jednoduchou plnou čáru.
- **Agregace** – Agregace reprezentuje vztah typu celek – část. Znázorňuje se jako jednoduchá plná čára, zakončená na jedné straně prázdným kosočtvercem. Ten je umístěn u té entity, která reprezentuje celek.
- **Kompozice** – Kompozice je podobná agregaci, avšak reprezentuje silnější vztah. Entita části nemá bez celku smysl. Pokud zanikne celek, zanikají automaticky i jeho části.
- **Generalizace** – Z hlediska implementace se jedná o dědičnost. Jedna entita dědí vlastnosti a chování jiné (ITnetwork, 2013).



Obrázek 9: Doménový model

Doménový model, znázorněný výše na obrázku č. 8, je v našem případě tvořen celkem 12 třídami. Třída Zaměstnanec je třídou s nejvíce vazbami, což je dáno tím, že právě zaměstnanci spravují celý systém. U všech těchto vazeb s třídou Zaměstnanec se jedná o asociační typ vazby.

Dále si můžeme všimnout, že model obsahuje také kompoziční vazby – příkladem je vazba mezi třídou Galerie a Fotka. Galerie v tomto případě představuje celek a jednotlivé fotky pak jeho součásti. Pokud zanikne galerie, pak zanikají také fotky, které byly součástí této galerie. Stejný vztah pojí také vazby Produkt – Galerie a Reference – Galerie. V obou těchto případech však vystupuje galerie jako součást připojovaná k celku.

V modelu nalezneme také dvě agregační vazby. Jedná se o vazby Materiál – Barva a Kalkulace – Box. Jak bylo uvedeno výše, agregační vazba je slabším typem vazby než je tomu u kompozice, což znemená, že pokud zanikne celek, nezanikají

automaticky všechny jeho součásti. Ku příkladu pokud zanikne některý z materiálů, barvy jež byly k němu přiřazeny nezaniknou, ale dále existují.

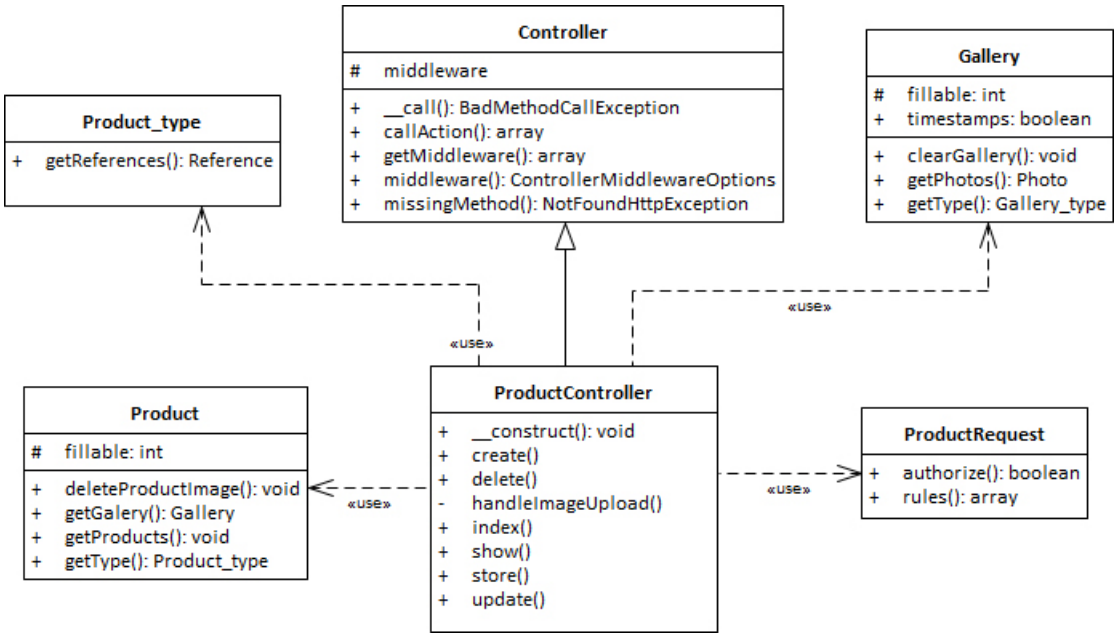
5 Návrh systému

Ve fázi návrhu systému vycházíme ze všech informací získaných během analýzy. Zatímco ve fázi analýzy jsme se zaměřovali na sběr požadavků zadavatele na systém a diagramy byly tvořeny z pohledu implementace v obecné rovině, výstupy návrhu by měly mít již konkrétní a skutečné vlastosti, které poslouží jako předloha pro pozdější implementaci.

5.1 Diagram tříd

Diagram tříd je diagram implementace. To je rozdíl oproti doménovému modelu, který byl spíše náčrt systému. Diagram tříd je již naostro, musí být úplný, když ho programátor přepíše do kódu, kód musí fungovat. Obsahuje tedy již všechny třídy, které program bude obsahovat. Třídy mají všechny atributy a také metody. Diagram je platformově závislý, tedy specifický pro určitý programovací jazyk (ITnetwork, 2013).

V této kapitole nebudeme uvádět kompletní diagram tříd pro celý systém a to z důvodu velkého rozsahu, nýbrž pouze fragment tříd, které obstarávají funkcionalitu pro tu část systému, která slouží pro správu produktů.

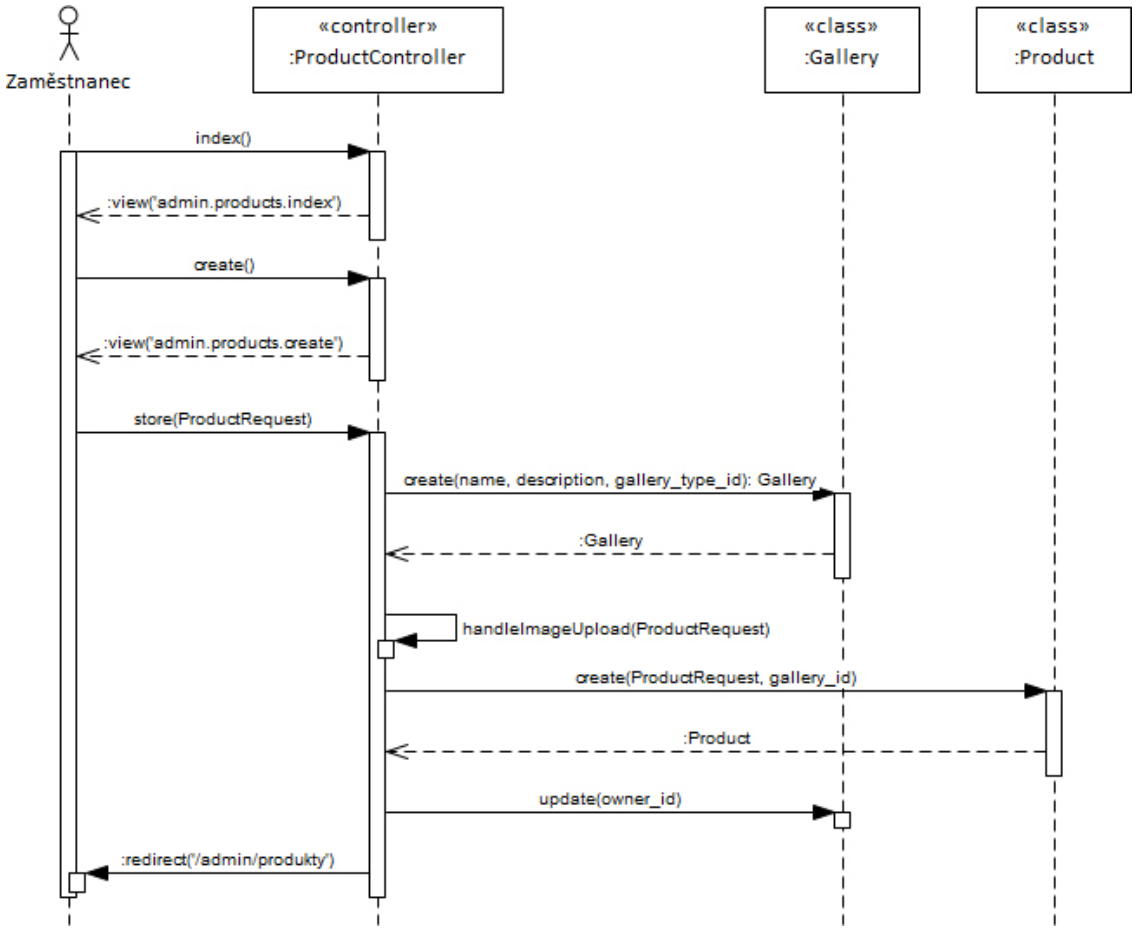


Obrázek 10: Diagram tříd pro správu produktů

Hlavní třídou je v tomto případě třída ProductController, která řídí veškeré požadavky pro manipulaci s produkty. Tato třída využívá také funkcionalitu několika jiných tříd. To je v diagramu znázorněno pomocí vazeb s přerušovanou čarou.

5.2 Sekvenční diagramy

Sekvenční diagramy zachycují časově uspořádanou posloupnost zasílání zpráv mezi objekty. Jednotlivé objekty zapojené do popisovaného případu užití jsou umístěn v horní části diagramu. Od nich pak vedou směrem dolů čáry (lifelines), které znázorňují běh času. Mezi čarami jsou pak zakresleny vodorovné šipky různých typů, které reprezentují zprávy posílané mezi objekty. Plné šipky značí volání, přerušované pak odpověď. Podlouhlé obdélníky na svislých čarách vyznačují dobu zpracovávání dané zprávy či čekání na odpověď (Wikipedia, 2016). Příkladem je následující sekvenční diagram pro vytvoření nového produktu.



Obrázek 11: Sekvenční diagram pro vytvoření nového produktu

5.3 Návrh databáze

Databáze byla navržena na základě požadavků firmy na funkcionalitu systému. Výsledné schéma je tvořeno celkem 20 tabulkami a je možné si jej prohlédnout v příloze A této práce.

Popis schéma databáze

základní kámen celého systému tvoří zaměstnanci, bez nich by existence systému neměla smysl. Zaměstnanci jsou ukládáni do tabulky *users* a u každého z nich je evidováno jeho jméno, email a přihlašovací údaje do systému. Heslo je ukládáno v zahešované podobě pomocí funkce Bcrypt. Rozlišení běžného zaměstnance a administrátora je realizováno na základě atributu *admin*.

Tabulkou s nejvíce vazbami je tabulka *calculations*, která slouží pro ukládání kalkulací. Tato tabulka ma také poměrně velké množství atributů, které jsou využity postupně v závislosti na aktuálním stavu kalkulace. Stav kalkulace je určen pomocí atributu *calculation_status_id*, který je zároveň cizím klíčem do tabulky *calculation_statuses*, jež uchovává textovou podobu jednotlivých stavů kalkulací. Dalším cizím klíčem je atribut *stable_id*. Tento atribut odkazuje do tabulky *stables*, kde je uložena vnitřní reprezentace boxů, které jsou předmětem kalkulace. Kalkulace také úzce souvisí s tabulkou *orders*, tedy s tabulkou zakázek. Zakázky vznikají ve chvíli kdy kalkulace přejdou do stavu *schválená*. Aby bylo zamezeno vzniku duplicitních dat, každá zakázka odkazuje pomocí cizího klíče *calculation_id* právě na tu kalkulaci, která vedla k jejímu vzniku. V neposlední řadě uvedené tabulky *calculations*, *stables* a *orders* mají vazbu v tabulce *dates*. Ta představuje funkci dimenzionální tabulky pro dimenzi datum a to z důvodu možnosti vytvářet přehledy výkonosti a různé statistiky.

Jednotlivé produkty jsou uloženy v tabulce *products*. Systém rozlišuje různé typy produktů – to je realizováno pomocí atributu *product_type_id*, což je cizí klíč do tabulky *product_types*. Každý produkt a také každá reference (tabulka *references*) má svou galerii (atribut *gallery_id*).

Pokud se podíváme blíže na tabulku *galleries*, zjistíme že má vazbu s tabulkami *gallery_types* a *photos*. První jmenovaná slouží pro identifikaci typu galerie, druhá pak pro ukládání fotek. Každá fotka má atribut *gallery_id*, který jednoznačně určuje, do které galerie patří.

Co se týče konfigurátoru boxů, pro jeho správnou funkci je potřeba spravovat materiály (tabulka *materials*) a barvy (tabulka *colors*). Mezi materiály a barvami platí vazba M:N, kterou je potřebo dekomponovat na dvě vazby typu 1:N. To je realizováno pomocí tabulky *color_material*. Stejně tak je potřeba rozložit vazbu mezi materiály a kategoriemi materiálů *materialcats*. Spojovací tabulkou je v tomto případě tabulka *material_materialcat*.

Zbývá zmínit ještě poslední dvě tabulky, kterými jsou *documents* a *testimonials*. Jak již jejich názvy napovídají, jedna slouží pro správu dokumentů, což není nic jiného než obsah jednotlivých webových stránek, druhá pak pro uchovávání názorů zákazníků.

5.4 Návrh 3D konfigurátoru

Významnou položku návrhu systému představoval 3D konfigurátor. Jednak bylo potřeba navrhnout jak bude implementován a pomocí jakých technologií, ale také například jak budou modely boxů ukládány nebo jak bude řešena správa konfigurátoru pomocí administrace.

Technologie

Při výběru technologie bylo potřeba brát v úvahu tyto požadavky:

1. Konfigurátor bude sloužit k vytváření 3D modelů venkovních boxů pro koně.
2. Bude umožňovat měnit parametry boxů.
3. Bude možné měnit materiály a barvy.
4. Uživatelé si budou moci ukládat své modely a následně opětovně upravovat.

Dále bylo potřeba zajistit, aby se konfigurátor rychle načítal, pracoval svižně, nebylo potřeba k jeho spuštění stahovat nějaké doplňky, ale také aby fungoval na co největším počtu zařízení a různých webových prohlížečích. S ohledem na všechny tyto požadavky byla jako vhodný prostředek pro implementaci zvolena JavaScriptová knihovna Three.js.

Tato knihovna je vhodná zejména pro svou širokou podporu webových prohlížečů a nezávislost na proprietárních pluginech ve webových prohlížečích. Dalším důvodem je také fakt, že využívá aplikačního rozhraní WebGL, které je v současné době široce podporováno (Can I use, 2016).

Ukládání modelů

K tomu aby bylo možné modely ukládat do databáze a opětovně je nahrávat, bylo nutné navrhnout vhodnou datovou strukturu. Vzhledem k tomu, že konfigurátor je naprogramován v jazyce JavaScript, bylo navrhnuto řešení ukládat modely v podobě JSON notace. To přináší hned několik výhod:

1. JSON je nezávislý na platformě.
2. Velmi snadné nahrávání z databáze, boxy jsou uloženy přímo ve formě JavaScriptového objektu.
3. JSON lze snadno převést na PHP objekt a naopak.
4. Není potřeba řešit proměnlivou strukturu boxů (například jednořadá stáj pro tři koně ma jiný počet boxů než dvouřadá stáj pro 10 koní).

Výsledná JSON struktura, která univerzálně popisuje libovolnou stáj byla vytvořena na základě reálných projektů. K tomuto účelu byly identifikovány společné

znaky různých venkovních boxů a ty byly zaneseny do výsledné struktury. Následující JSON reprezentuje dvouřadé venkovní boxy pro 4 koně:

```
{
  "type_id": 1,
  "type": "double_row",
  "unit_count": 4,
  "wall_height": 2500,
  "front_wall": { "type": 1, "door": 1 },
  "rear_wall": { "type": 1, "door": 0 },
  "material": { "planks": { "material": 7, "color": 0 } ... },
  "paddock": { "left": 4000, "right": 4000 },
  "row_length": { "left": 3500, "right": 3500 },
  "room": {
    "1": {
      "id": 1,
      "type": 1,
      "next": 2,
      "prev": null,
      "origin_distance": 0,
      "front_wall": 1,
      "rear_wall": 2,
      "left_wall": null,
      "right_wall": 1,
      "paddock": 1,
      "width": 3500
    }
    ...
  },
  "side_wall": {
      "1": {
      "type": 1,
      "side": "left"
    }
    ...
  },
  "corridor_width": 3000,
  "width": 10000,
  "length": 7000,
  "gable_height": 1400,
  "roof": { "type": 1, "overlap_x": 500, "overlap_z": 500 }
}
```

Výše uvedený JSON uchovává informace jako jsou typ stáje (*type*), počet boxů (*unit_count*), výšku stěny (*wall_height*), informace o střeše (*roof*) apod. Informace o jednotlivých boxech nalezneme pod indexem (*room*). Zde je každý box definován pomocí několika informací:

- **id** – Identifikátor boxu.
- **type** – Typ boxu.

- **next** – Identifikátor následujícího boxu.
- **prev** – Identifikátor předchozího boxu.
- **origin_distance** – Vzdálenost od počátku souřadného systému.
- **front_wall** – Typ přední stěny boxu.
- **rear_wall** – Typ zadní stěny.
- **left_wall** – Identifikátor levé boční stěny.
- **right_wall** – Identifikátor pravé boční stěny.
- **paddock** – Informace o tom, zda je k boxu připojen paddock (1 = ano, 0 = ne).
- **width** – Šířka boxu.

Informace o použitých materiálech se nachází pod indexem material. Pro každou součást či skupiny součástí, u kterých lze měnit materiál a barvu, uchováváme tyto informace pomocí identifikátoru materiálu a identifikátoru barvy. Ve výše uvedené ukázce můžeme vyčíst, že pro dřevěné výplně (planks) je použit materiál s id 7 a žádná barva (id 0). Celkem je potřeba uchovávat informace pro 11 skupin součástí.

5.5 Návrh grafického designu

Grafický design systému byl navrhován tak, aby byl v první řadě přehledný a neodrazoval návštěvníky od jeho používání. Dále bylo nutné přihlédnout na požadavky firmy, mezi které patřilo například to, že web by měl být responzivní. Proces tohoto návrhu probíhal v následujících bodech:

1. Seznámení se strategií firmy a účelem webu.
2. Vytvoření wireframu.
3. Vytvoření grafického návrhu.

Strategie firmy

Aby byl vytvořen správný koncept webdesignu, je potřeba pochopit strategii podnikání. Jaké jsou cíle, pro koho bude web určen, co by měl návštěvník provést za akci, jaké informace na webu nalezne atd (PixelPerfect, 2015).

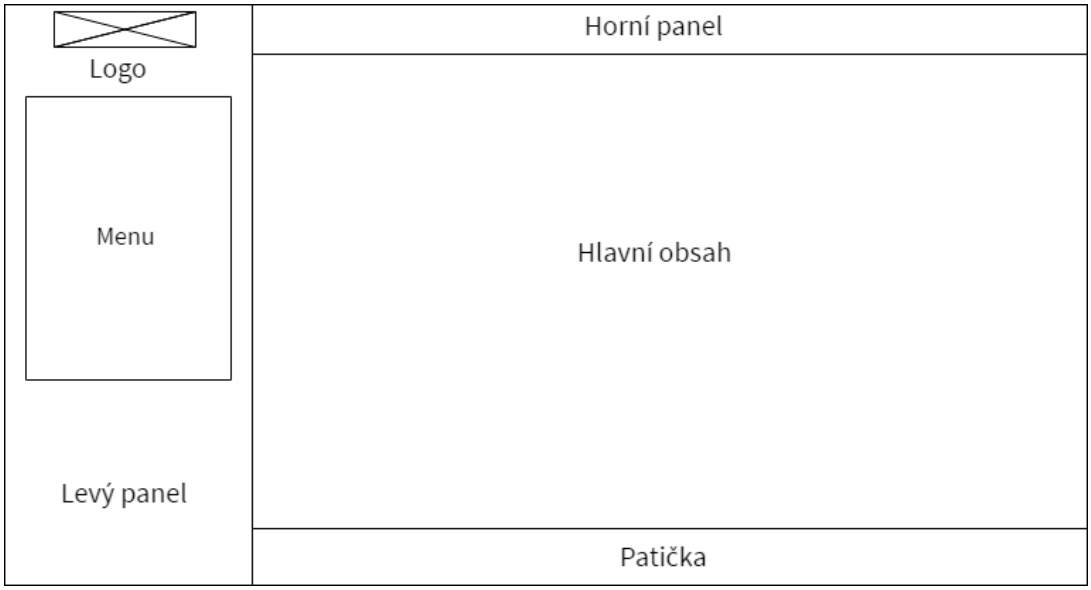
V případě firmy Metal Ryspol, s. r. o. je účelem webu získání nových zákazníků. Web by měl uživatele vést k provedení akcí, které požadujeme. Tím je na mysli vyvolání zájmu o produkty firmy a snahy dozvědět se více (například zjistit cenu).

Wireframe

Jakmile víme za jakým účelem má být web vytvořen, můžeme začít pracovat na konrétním designu. První fází je obvykle návrh toho jaké prvky bude webová strán-

ka obsahovat a také jak budou na ní rozmístěny. K tomuto účelu se vytváří tzv. wireframy nebo-li drátěné modely. Wireframy jsou tvořeny pouze pomocí čar a textu a neobsahují žádné obrázky či jinou grafiku. Jejich účelem je najít nejvhodnější rozmístění prvků.

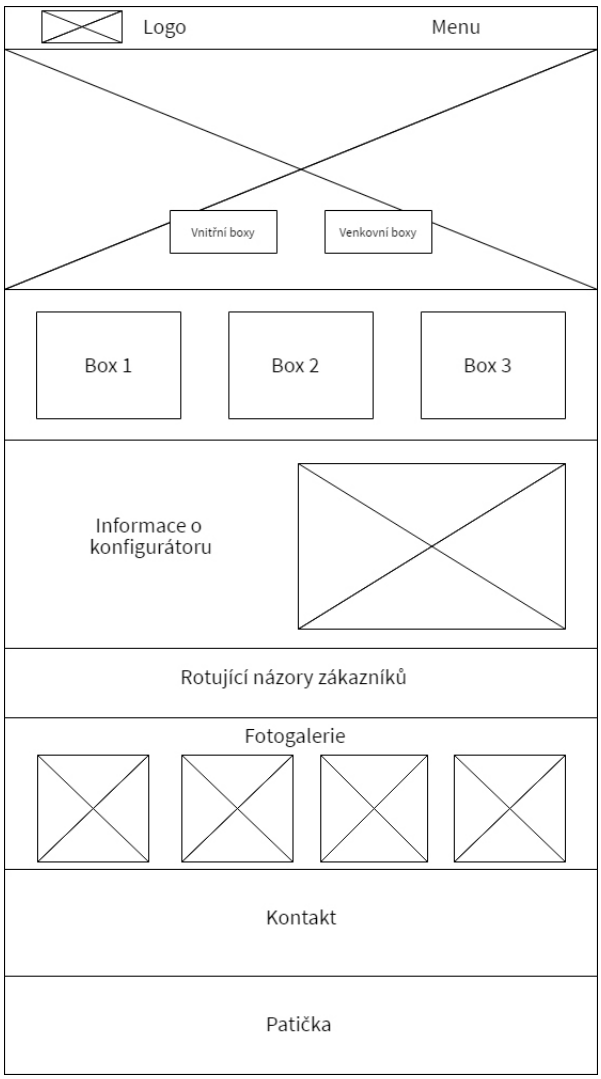
V našem případě bylo potřeba vytvořit dva různé wireframy, jeden pro veřejnou část systému a druhý pro administraci. Nejprve si představme wireframe pro administraci.



Obrázek 12: Wireframe administrace

Rozložení prvků v administraci je navrženo tak, aby bylo přehledné a intuitivní. Hlavní menu se nachází v levém postraním panelu a tuto pozici ani strukturu nikdy nemění. To stejné platí i pro horní panel a patičku. Jediným prvkem, který se mění je obsah a to v závislosti na konkrétní podstránce.

Wireframe pro domovskou stránku veřejné části je již o něco složitější, přesto i zde platí stejné zásady jako u administrace. Hlavní menu se zde nachází v horním panelu a je vodorovné. Primárním prvkem hlavní strany je kontejner s velkou fotografií, který slouží jako směrovník. Pod tímto kontejner se nachází již menší boxy s informacemi, které by návštěvníky mohly nejvíce zajímat. Dálšími prvky jsou kontejner s informacemi o designeru boxů, ukázka fotek z fotogalerie, kontakt a patička. Rozmístění těchto prvků demonstruje následující obrázek.



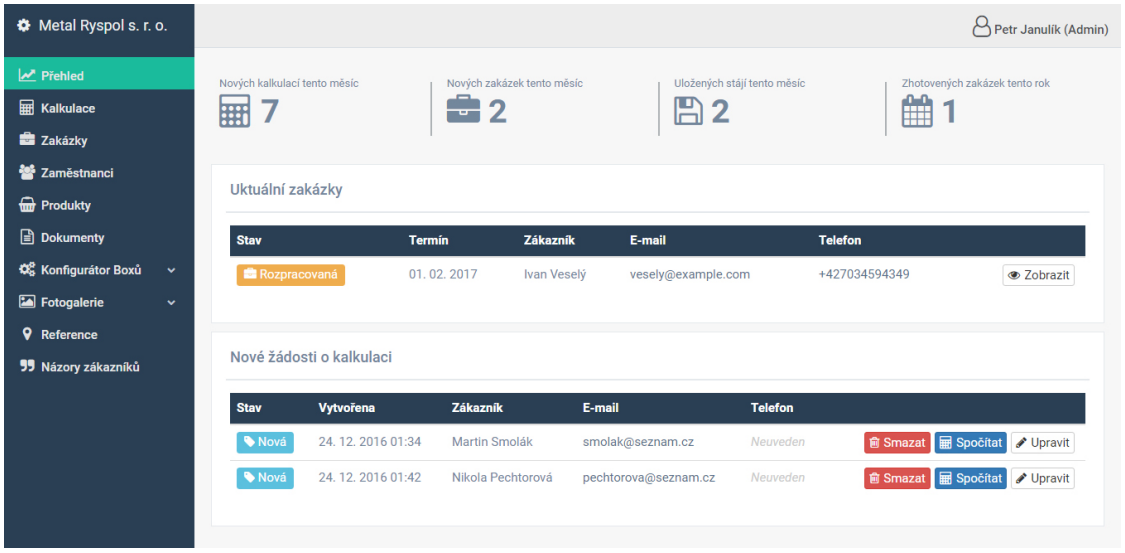
Obrázek 13: Wireframe domovské stránky veřejné části systému

Grafický návrh

Konečná podoba grafického návrhu vychází z předchozích wireframů. Vzhledem k tomu, že administrace slouží pouze pro vnitřní potřeby firmy, není nutné vytvářet unikátní styly a proto je vzhled jednotlivých prvků založen na frameworku Bootstrap. Domovská stránka administrace se nachází na níže uvedeném obrázku.

Můžeme si všimnout, že položky hlavního menu, které se nachází v levém postraním panelu, jsou tvořeny jednak textem, ale také jednoduchými ikonami, které vystihují jejich význam a pomáhají se tak lépe orientovat v poměrně dlouhém seznamu. Tyto ikonky jsou rovněž použity u tlačítek. V horním panelu můžeme najít informaci o právě přihlášeném uživateli, zároveň se zde nachází uživatelské menu s možností odhlásit se nebo upravit svůj profil.

Obsah stránky vyjadřuje aktuální přehled a zobrazuje nejdůležitější informace. Patří mezi ně seznam aktuálních zakázek, kde jsou zobrazeny všechy rozpracované zakázky a jejich termíny, dále seznam nových žádostí o kalkulaci a také několik statistických ukazatelů.



Obrázek 14: Grafický návrh administrace

Při návrhu grafického designu pro veřejnou část systému, jsme se již nemohli spokojit pouze s tuctovým vzhledem prvků, ale naopak bylo usilováno o originalitu, která by jasně definovala firmu. Barevné schéma vychází ze současného webu firmy. Primární barvou je oranžová, která je doplněna o několik neutrálních šedých odstínů.

Jednotlivé prvky jsou navrženy tak, aby nezahlcovali návštěvníka velkým množstvím informací najednou. To je zajištěno pomocí jejich větší velikosti, dále krátkými a výstižnými texty a v neposlední řadě také velikostí těchto textů. Aby byla práce s webem pohodlná horní panel s hlavní nabídkou má fixní polohu – to znamená, že zůstává na stejném místě i v případě, že uživatel scrolluje dolů. Odpadá tak nutnost hledat menu, když se nacházíme ve spodní části delší stránky.

6 Implementace

V této kapitole je popsána implemetnace klíčových částí systému. Kapitola je rozdělena na tři podkapitoly, kde první je zaměřena na administraci, druhá na konfigurátor boxů a poslední se zabývá implementací veřejné části systému.

Jak již bylo zmíněno ve druhé kapitole této práce, implementace systému je postavena na PHP frameworku Laravel. Samotné programovací práce byly realizovány pomocí vývojového prostředí Sublime Text 3.

6.1 Implementace administrace

Administrace tvoří součást systému, která není dostupná běžným návštěvníkům webu, slouží výhradně zaměstnacům firmy pro správu obsahu veřejné části a vyřízování žádostí o kalkulaci.

Použité knihovny

Administrace je vzhledem k povaze jednotlivých úkonů poměrně komplexní a vyžaduje celou řadu různých prvků a funkcí pro vykonávání těchto úkonů. Cílem bylo poskytnout co nejjednoduší manipulaci se systémem, tak aby uživatel nemusel až příliš přemýšlet nad elementárními úkony, jako je například zadání data ve správném formátu. K tomu nám dopomáhá celá řada knihoven třetích stran uvedených v následujícím seznamu.

- **Bootstrap** – HTML a CSS framework, modální okna.
- **Font Awesome** – Sada více jak 600 různých ikon ve formě fontu.
- **Bootstrap Colorpicker 2** – Knihovna poskytující prvek pro snadný výběr barvy a zadání barvy v hexadecimálním formátu.
- **DataTables** – Knihovna použitá pro tabulky, umožňuje řadit obsah, filtrovat, stránkovat apod.
- **Bootstrap-datepicker** – Knihovna poskytující prvek pro snadný výběr data z kalendáře.
- **DropzoneJS** – Knihovna pro správce nahrávání obrázků.
- **Parsley** – Validace formulářů na straně klienta.
- **Ion.RangeSlider** – Knihovna pro snadné zadávání číselných hodnot z rozsahu v podobě posuvníků.
- **Select2** – Knihovna pro uživatelsky příjemné výběrové prvky.
- **Starrr** – Knihovna pro hodnocení v podobě hvězdiček.
- **TinyMCE** – Wysiwyg editor.

Přístup do systému

Do administrace se přistupuje přes adresář `/admin`. Pro přístup do této sekce a jednotlivých podstránek je potřeba se prokázat platným uživatelským jménem a heslem prostřednictvím formuláře, který se zobrazí.

Tomuto procesu se říká autentizace a je zajištěna pomocí vestavěného autentizačního systému frameworku Laravel. Pomocí tzv. směrování (routing) definujeme, u kterých URL adres vyžadujeme autentitaci, viz následující ukázka kódu.

```
1  //Prihlaseni
2  Route::get('login', 'Auth\LoginController@showLoginForm');
3
4  //Routy pro administraci
5  Route::group(['prefix' => 'admin/', 'middleware' => 'auth'],
       function () {
6
7      //Dashboard
8      Route::get('/', 'Admin\AdminController@index');
9
10     //Uzivatelsky profil
11     Route::get('/profil', 'Admin\UserController@showProfile');
12     Route::patch('/profil', 'Admin\UserController@updateProfile');
13     Route::get('/zmena-hesla', 'Admin\UserController@showPassword');
14     Route::patch('/zmena-hesla', 'Admin\
       UserController@updatePassword');
15
16     ...
17
18 });
```

Stránka s formulářem pro přihlášení (`/login`) je logicky veřejná, dále definujeme skupinu cest s prefixem (`/admin`). Této skupině je přiřazena tzv. *middleware*, která zajišťuje kontrolu zda je uživatel přihlášen, při každém pokusu o zobrazení obsahu pomocí jedné z cest spadajících do skupiny.

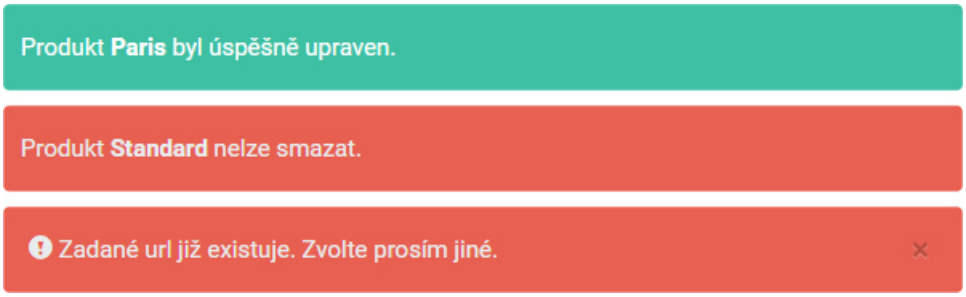
Správa uživatelských účtů

V systému vystupují dva typy uživatelů – běžný uživatel a administrátor. Systém neposkytuje žádnou veřejnou registraci, nové uživatelské účty vytváří administrátor pomocí administrace. Přidání nového zaměstnance tedy probíhá tak, že nejprve administrátor vytvoří uživatelský účet, poté poskytne přihlašovací údaje novému zaměstnanci, ten si tyto údaje po prvním přihlášení změní podle sebe.

Administrátor má dále právo měnit veškeré uživatelské účty a jmenovat nové administrátory. Přístup do správy uživatelů se nachází v hlavním menu a je viditelný pouze pro administrátory, pro běžné uživatele je skrytý.

Notifikace a modální okna

Pro celou administraci platí, že systém poskytuje odezvu na operace, které uživatel provedl a informuje jej o jejich úspěšném provedení či chybě, která nastala.



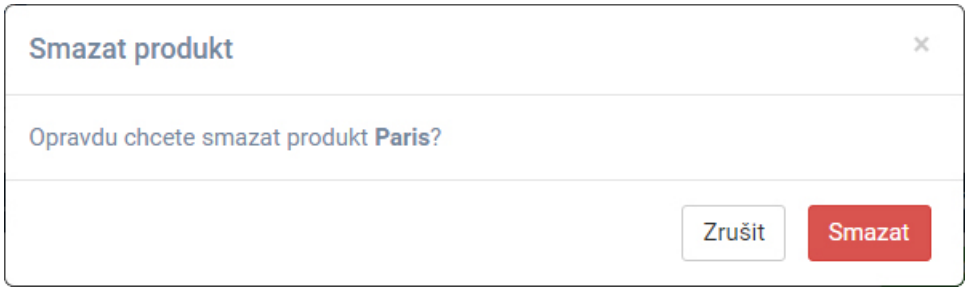
Obrázek 15: Ukázka systémových notifikací

Výše uvedený obrázek je ukázkou různých druhů notifikací, se kterými se může uživatel systému setkat. Zelenou barvou jsou zobrazeny různé krátké zprávy informující o úspěšném provedení akce, červenou pak zpávy informující o chybách. Chybové zprávy dále rozlišujeme na méně důležité a důležité. Méně důležité chybové zprávy, stejně jako zprávy o úspěchu, po 5 vteřinách mizí, zatímco důležité zůstavají zobrazeny dokud je uživatel nenechá zmizet pomocí ikony křížku.

```
1 $('div.alert').not('div.alert-important').delay(5000).slideUp(500);
```

Příkladem, kdy je nežádoucí samovolné mizení chybových hlášek je například při validaci formulářů. V takovém případě může být totiž takových hlášek zobrazeno více najednou a uživatel nemusí vše stihnout přečíst za uvedených 5 vteřin.

Další způsob jakým systém komunikuje s uživatelem představují modální okna. Ty jsou zobrazeny ve chvíli, kdy se uživatel chystá provést nějakou nenávratnou akci – například něco smazat. Účelem modálních oken je předcházet různým nechtěným překlikům a problémům s nimi spojenými. Proto před každým provedením nenávratné akce, systém ověří, zda uživatel tuto akci chce opravdu provést.

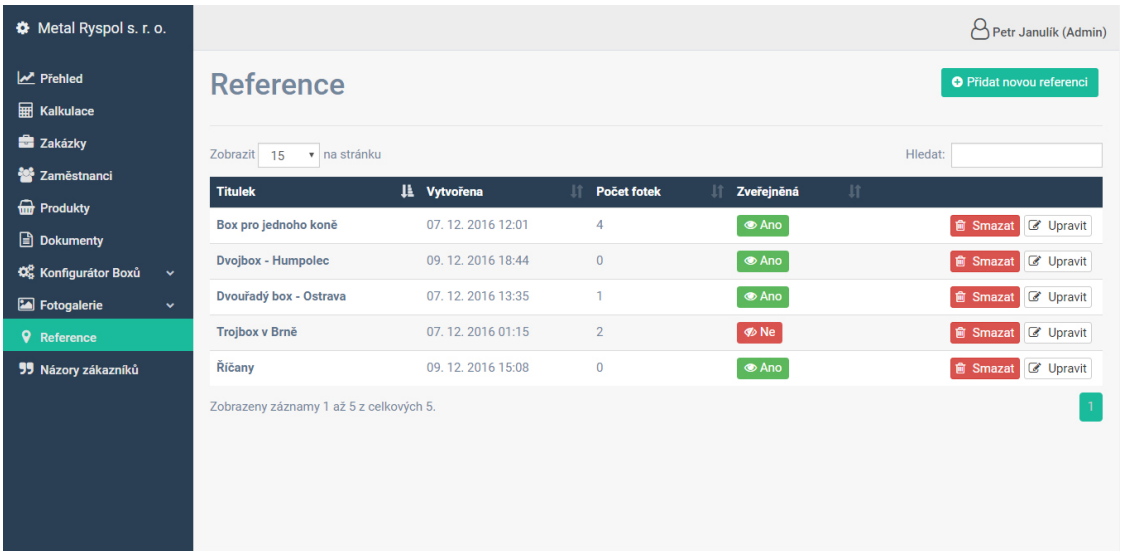


Obrázek 16: Ukázka modálního okna

Správa webového obsahu

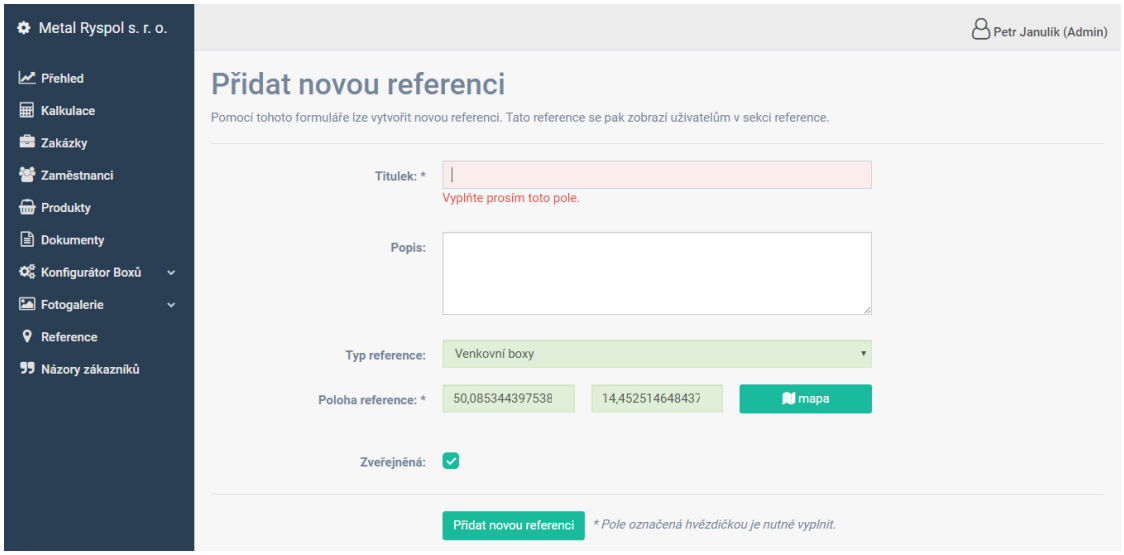
Správou webového obsahu máme na mysli správu jednotlivých článků, fotogalerií, produktů, referencí a zákaznických názorů. Každá z těchto sekcí má svůj odkaz v hlavním menu administrace.

Při vstupu do libovolné z výše uvedených sekcí se uživateli zobrazí výpis položek dané sekce ve formě tabulky. Pro každou položku jsou v tabulce uvedeny ty nejdůležitější informace, jako například zda je zveřejněna. V posledním sloupci se vždy nachází tlačítka pro operace, které lze s danou položkou provádět. Výpis je dále možné filtrovat za pomoci vyhledávacího pole, a také řadit podle jednotlivých sloupců tabulky. Uživatel má také možnost měnit počet položek zobrazených na stránku.



Obrázek 17: Správa referencí

Vytváření a editace nových položek je realizována pomocí příslušných formulářů pro daný typ položky. Následující obrázek zobrazuje formulář pro přidání nové reference.



Obrázek 18: Formulář pro vytvoření nové reference

Veškeré formuláře v systému prochází dvojí validací – na straně klienta a na straně serveru. Validaci na straně klienta obstarává JavaScriptová knihovna Parsley.js. Tento způsob validací je výhodný v tom, že nedovolí uživateli odeslet formulář dokud nejsou všechny jeho pole správně vyplněny. Druhý stupeň validace probíhá již na straně serveru.

Pro každý typ požadavku na vytvoření či editaci existuje speciální třída, kde jsou definována pravidla pro vstupní data. Následující ukázka kódu obsahuje třídu *ReferenceRequest*, která slouží pro kontrolu vstupních dat při editaci či vytváření nové reference.

```
1  class ReferenceRequest extends FormRequest{
2
3      public function authorize(){
4          return true;
5      }
6
7      public function rules(){
8          return [
9              'title' => 'string|required',
10             'description' => 'string|nullable',
11             'latitude' => 'numeric|required',
12             'longtitude' => 'numeric|required',
13             'product_type_id' => 'integer|exists:product_types,id|
     required',
14             'published' => 'integer|in:0,1'
15         ];
16     }
17
18 }
```

Součástí této třídy je metoda *rules()*, která vrací sadu pravidel pro daný požadavek ve formě pole. Tato metoda je volána při každém požadavku na změnu nebo vytvoření nové reference. Pravidla převezme validátor a provede kontrolu vstupu. V případě, že některé z pravidel neprojde testem, požadavek je zamítnut a dojde k přesměrování zpět na stránku s formulářem včetně vypsání k jakým chybám došlo.

Správa konfigurátoru boxů

Součástí administrace je také správa konfigurátoru boxů. Tato položka zahrnuje kontrolu nad tím jaké materiály a v jakých barevných variacích jsou uživatelům v konfigurátoru k dispozici.

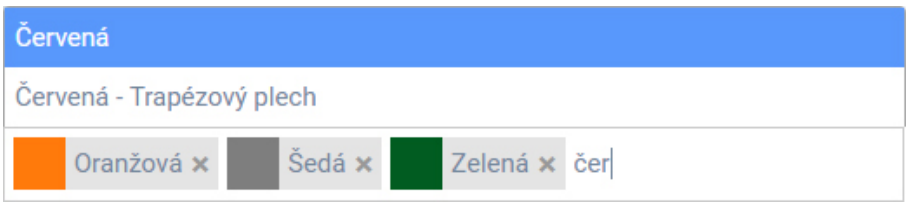
Administrace poskytuje zvlášť správu pro barevné odstíny a zvlášť správu materiálů. V obou případech je, při vytváření nových položek, potřeba vycházet z knihovny Three.js, na které je konfigurátor postaven. U materiálů je potřeba definovat tyto vlastnosti:

- textura a její rozměry
- způsob mapování textury na objekty
- neprůhlednost materiálu
- barva odlesků
- intenzita lesklosti materiálu
- výchozí barva materiálu

V případě barev je pak nutné specifikovat pouze barevný odstín a barvu odlesků včetně intenzity. Při manipulaci s těmity hodnotami je usilováno o jednoduchost, tak aby i nezkušený uživatel mohl snadno vytvořit nový materiál či barvu. To je zajištěno pomocí vhodných formulářových prvků, které výrazně usnadňují práci při vkládání nestandardních hodnot jako je hexadecimální kód barvy.

Tento přístup kdy jsou barvy a materiály vytvářeny zvlášť, je výhodný v tom, že se vyhneme vzniku duplicit materiálů, které se lyší pouze svým barevným odstínem. Je však potřeba zajistit propojení mezi materiály a příslušnými barvami.

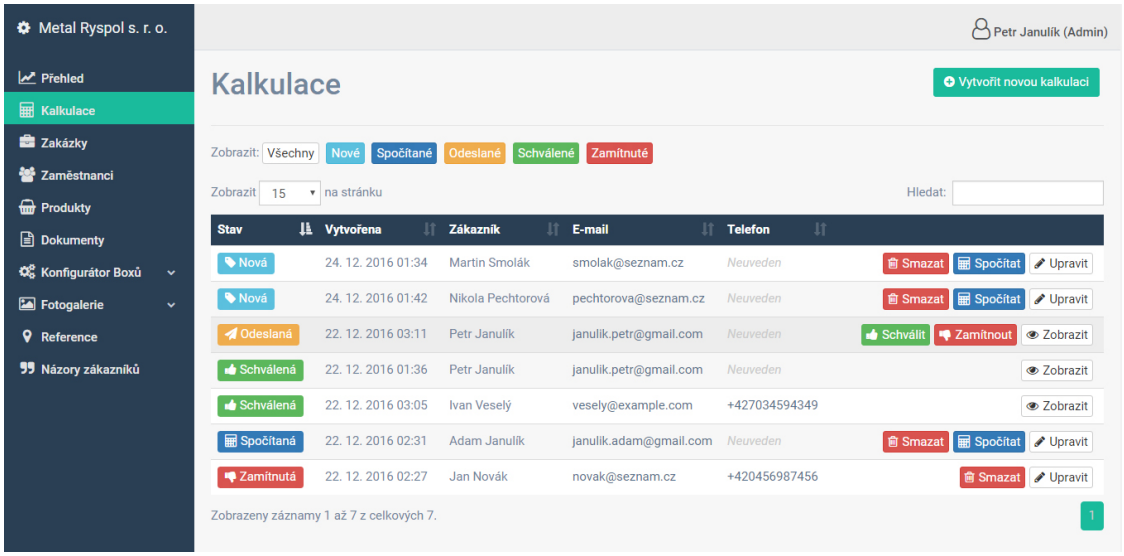
Z hlediska uživatelského rozhraní je tento problém řešen pomocí rozbalovací nabídky s možnostní vícenásobné volby. Vzhledem k tomu, že použití tohoto prvku ve své původní podobě není zrovna uživatelsky přívětivé, byla za pomoci knihovny Select2 vytvořena vlastní nabídka, tak jak je uvedeno na následujícím obrázku.



Obrázek 19: Formulářový prvek pro výběr barev

Správa kalkulací a zakázek

Další užitečnou funkcí, kterou administrace poskytuje je správa kalkulací. Ta je navržena pro firmu za účelem úspory času, součástí je totiž speciální kalkulátor celkových nákladů, který umožňuje mnohonásobně rychleji spočítat celkovou cenu boxů.



Obrázek 20: Stránka s přehledem všech kalkulací

Po vstupu do správy kalkulací se zobrazí přehled všech kalkulací. Na obrázku č. 20 můžeme vidět, že každá z nich má přiřazen určitý stav, od kterého se odvíjejí také akce, které lze s danou kalkulací provádět. Jednotlivé stavy, kterých mohou kalkulace během svého životního cyklu nabývat, jsou uvedeny v následující tabulce.

Tabulka 2: Tabulka možných stavů kalkulací

Stav	**Možné akce**
Nová	smazat, spočítat, upravit
Spočítaná	smazat, spočítat, upravit
Odeslaná	schválit, zamítnout, zobrazit
Schválená	zobrazit
Zamítnutá	smazat, upravit

Každou novou kalkulaci je možné buď smazat, upravit nebo spočítat. Spočítáním je myšleno vyčíslení celkových nákladů a stanovení konečné ceny pro zákazníka. Spočítanou kalkulaci je následně možné odeslat zákazníkovi (stav odeslaná) nebo uložit pro pozdější odeslání (stav spočítaná). Odesláním kalkulace je také znemožněno její další upravování a smazání. Dle reakce zákazníka na cenovou nabídku pak zaměstnanec danou kalkulaci označí buď jako schválenou nebo jako zamítnutou. Zamítnuté lze znovu upravit, čímž se dostanou zpět do stavu nová a celý proces se opakuje. Schválením kalkulace vznikne nová zakázka v sekci zakázky.

Výpočet kalkulace je realizován kalkulátorem, který je zobrazen po kliknutí na volbu spočítat. Součástí kalkulátoru je přehled jednotlivých položek kalkulace, kde je uvedeno jaký typ boxů zákazník požaduje a také detailní výpis všech součástí včetně rozměrů. Je možné také zobrazit 3D model stáje, pokud se jedná o venkovní boxy.

Princip výpočtu celkových nákladů je založen na vyčíslení celkové potřeby materiálů a barev, jež jsou poté vynásobeny svými cenami. Tento výpočet je proveden pro jednotlivé části boxů zvlášť, tak jak uvádí následující obrázek.

Obrázek 21: Ukázka výpočtu nákladů

Výše uvedená tabulka je určena pouze pro potřeby demonstrace a není proto úplná. Každý materiál a barva má svou cenu za danou jednotku. Tuto cenu je však možné stanovovat individuálně, pomocí příslušných polí. Veškeré provedené změny způsobí automatické přepočítání a aktualizaci celkové sumy.

Pod tabulkou s rozpisem celkových nákladů se nachází souhrnná tabulka kalkulace. Zde jsou vyčísleny celkové náklady na potřebný materiál a také se tu nachází

tři vstupní pole, kde je možné stanovit výši ostatních nákladů, účtované práce a nákladů na dopravu. Ná základě těchto informací je stanovena celková cena bez DPH a z ní následně cena s DPH.

Obrázek 22: Ukázka výpočtu nákladů

K takto vypočtené kalkulaci lze přidat poznámku a poslat ji zákazníkovi prostřednictvím e-mailu.

6.2 Implementace konfigurátoru boxů

Konfigurátor boxů je naprogramován pomocí JavaScriptové knihovny Three.js. Na základě této knihovny byl vytvořen vlastní modul nazvaný StableBuilder.js, který poskytuje strukturu reprezentující stáj a veškeré editační operace nad toutu strukturou včetně vykreslování modelu do prohlížeče.

Scéna

Základním prvkem konfigurátoru je objekt *THREE.Scene*, který představuje scénu. Scénu je na počátku potřeba nakonfigurovat, vložit do ní kameru a světelné zdroje.

```
1  //scena
2  PRIV.scene = new THREE.Scene();
3
4  //kamera
5  PRIV.camera = new THREE.PerspectiveCamera( 30, window.innerWidth /
       window.innerHeight, 300, 100000 );
6
7  //osvetleni
8  PRIV.scene.add( new THREE.AmbientLight( "#312f16" ) );
9  PRIV.scene.add( new THREE.HemisphereLight( "#ffffbb", "#080820", 1 )
       );
10
11 var light = new THREE.DirectionalLight( "#ffffff", 0.8 );
12     light.position.set( 20000, 3000, 20000 );
13 PRIV.scene.add( light );
```

```
14
15 var light2 = new THREE.DirectionalLight( "#ffffff", 0.2 );
16     light2.position.set( 0, 3000, 20000 );
17 PRIV.scene.add( light2 );
18
19 var light3 = new THREE.DirectionalLight( "#ffffff", 0.8 );
20     light3.position.set( -20000, 3000, -20000 );
21 PRIV.scene.add( light3 );
22
23 var light4 = new THREE.DirectionalLight( "#ffffff", 0.2 );
24     light4.position.set( 0, 3000, -20000 );
25 PRIV.scene.add( light4 );
```

Ve scéně se vyskytuje jedna hlavní kamera a celkem 6 světelných zdrojů. Jednotlivé světelné zdroje tvoří ambientní světlo, což je světlo přicházející ze všech směrů se stejnou intenzitou, dále hemisferické světlo, které je speciálním typem ambientního světla definující barvu světla přicházející od zemského povrchu a barvu světla oblohy a také 4 přímé světla. Přímé světla simulují slunce, jedná se o nekonečně vzdálený zdroj, ze kterého se světlo šíří jedním směrem.

V našem případě máme 2 přímé světla s vysokou intenzitou svitu a dvě s nízkou intenzitou. Světla s vysokou intenzitou jsou umístěny v protilehlých rozích scény, tak aby byly osvíceny všechny 4 strany stáje stejnou intenzitou světla. Světla s nižší intenzitou jsou pak umístěny před a za stájí a vytváří efekt zvýraznění hran.

Objekty

Veškeré objekty, které chceme zobrazit je potřeba umístit do scény. Každý objekt má svou velikost, umístění a otočení vzhledem k jednotlivým osám. Dále je možné definovat vlastnosti jako neprůsvitnost, lesklost, barva, textura apod. Samotné modelování objektů je realizováno za pomoci skládání geometrických primitiv (bod, úsečka, plocha) ve složitější objekty. Knihovna Three.js nám ulehčuje práci v tom, že poskytuje funkce pro snadné vytvoření základních geometrických těles jako kvádr, koule, válec, jehlan a další.

Následující ukázka představuje funkci pro vytvoření ocelového profilu tzv. jeklu, který je nedílnou součástí každé stěny stáje.

```
1 PRIV.composeJekl = function(width, length, material){
2
3     var jekl_material = PRIV.getMaterial(material, length, width);
4     var jekl = new THREE.Mesh(new THREE.BoxGeometry(length, width,
     width), jekl_material);
5         jekl.position.y = width/2;
6         jekl.position.z = -width/2;
7     return jekl;
8
9 };
```

Vzhledem k tomu, že pro postavení celé stáje je potřeba velké množství jeklů s různými délkami a povrchovými úpravami, funkce přebírá v parametru tyto infor-

mace a podle nich vytvoří a vrátí jekl s požadovanými rozměry a materiálem. Na tomto principu je založen celý proces výstavby stáje. Všechny složité objekty jako stěny, okna nebo dveře se skládájí z jednodušších objektů.

Constructive Solid Geometry

V případě, že chceme vytvořit některé atypické objekty, nemusí nám prosté skládání objektů do sebe stačit nebo to může být až příliš složité. V takovém případě je vhodné využít operací CSG (Constructive Solid Geometry). Tyto operace poskytuje malá knihovnička ThreeCSG.js

Constructive solid geometry je vektorové modelování geometrických objektů. Tyto objekty se konstruují z primitivních geometrických těles (koule, kvádr, válec, kužel, toroid apod.) operacemi sjednocení, průniku a rozdílu (Wikipedia, 2016).

```
1  PRIV.composeU = function(width, height){
2
3      var material = PRIV.getMaterial("steel_outside", width, height);
4
5      var a = new THREE.Mesh(new THREE.BoxGeometry(width, height,
       width), material);
6          a.position.y = height/2;
7
8      var b = new THREE.Mesh(new THREE.BoxGeometry(width-10, height,
       width), material);
9          b.position.y = height/2;
10         b.position.z = 5;
11
12     var a_BSP = new ThreeBSP(a);
13     var b_BSP = new ThreeBSP(b);
14     var subtract_bsp = a_BSP.subtract(b_BSP);
15
16     var U = subtract_bsp.toMesh(material);
17         U.position.z = -width/2;
18
19     return U;
20
21 };
```

Výše uvedená funkce slouží pro vytvoření ocelového profilu U. Profil je vytvořen za pomoci dvou kvádrů, kde jeden z nich má menší rozměry. Vhodným umístěním a poté provedením operace rozdíl vznikne požadovaný profil.

Princip činnosti konfigurátoru

Princip činnosti konfigurátoru byl nastíněn již v kapitole týkající se návrhu. Základem je JSON struktura, která definuje stáj. Ná počátku je tedy potřeba tuto strukturu vygenerovat, což se děje poté co uživatel specifikuje jaký typ stáje a kolik boxů požaduje. Zároveň dojde k vykreslení specifikované stáje.

Následně uživatel může stáj různě modifikovat pomocí postraních nabídek nástrojů. Při každém volání některé z akcí je nejprve vytvořena nová změněná součást, poté je odstraněna původní a nahrazena novou. Zároveň dojde ke změně JSON struktury a překreslení modelu.

Ukládání vytvořených stájí je realizováno pomocí ajaxové funkce, která uloží do databáze JSON reprezentující stáj. Uživatel není zatěžován žádnou registrací, pro uložení stačí zadat e-mail. Systém vygeneruje pro každou uloženou stáj 2 unikátní hash kódy o délce 8 znaků, které slouží pro pozdější zobrazení a editaci stáje. Tyto kódy jsou součástí potvrzovacího e-mailu ve formě 2 url adres, kde jedna slouží pro přístup do editačního módu, druhá pak do módu zobrazovacího, kde není možné stáj editovat. Tento odkaz lze využít například pro různé sdílení na sociálních sítích apod.

6.3 Implementace veřejné části systému

Ačkoliv je administrace srdcem celého systému a pro správné fungování webu je naprosto nepostradatelná, veřejná část systému je ta, na které opravdu záleží, je to ta část, která prezentuje firmu, ta která má za úkol získat nové zákazníky a přinést firmě úspěch.

S ohledem na tuto skutečnost byla provedena implementace veřejné části i v našem případě. Implementovaný web se snaží držet krok s moderními trendy v oblasti webového designu, ale zároveň chce poskytnout informace v účelné a přehledné podobě, tak aby návštěvníci našli to co hledají a také aby naplňoval účel, ke kterému byl určen. V této podkapitole jsou popsány vybrané části veřejné části tohoto systému.

Úvodní strana

Úvodní strana je velmi často tou první stranou, kterou návštěvník shlédne jako první. Proto je důležité aby zanechala dobrý dojem. Úvodní strana nového webu byla implementována tak, aby sloužila jako výkladní skříň firmy. Ukázku lze najít v příloze B.

Vstupním bodem úvodní strany je velký kontejner, který slouží jako rozcestník mezi vnitřními a venkovními boxy. Dá se totiž očekávat, že návštěvníci budou přicházet na web za účelem zájmu o boxy pro koně a úkolem tohoto rozcestníku je, aby je navedl správným směrem. Pod rozcestníkem se nachází menší boxy s dalšími důležitými informacemi.

Dálšími prvky úvodní strany jsou kontejner, který upozorňuje na možnost vytvořit si modely boxů pomocí konfigurátoru, ale také několik náhodně vybraných fotek z fotogalerie a karusel, ve kterém rotují ohlasy spokojených zákazníků (tzv. testimonials). Úplně dole se pak nachází základní kontaktní informace.

Podání žádosti o kalkulaci

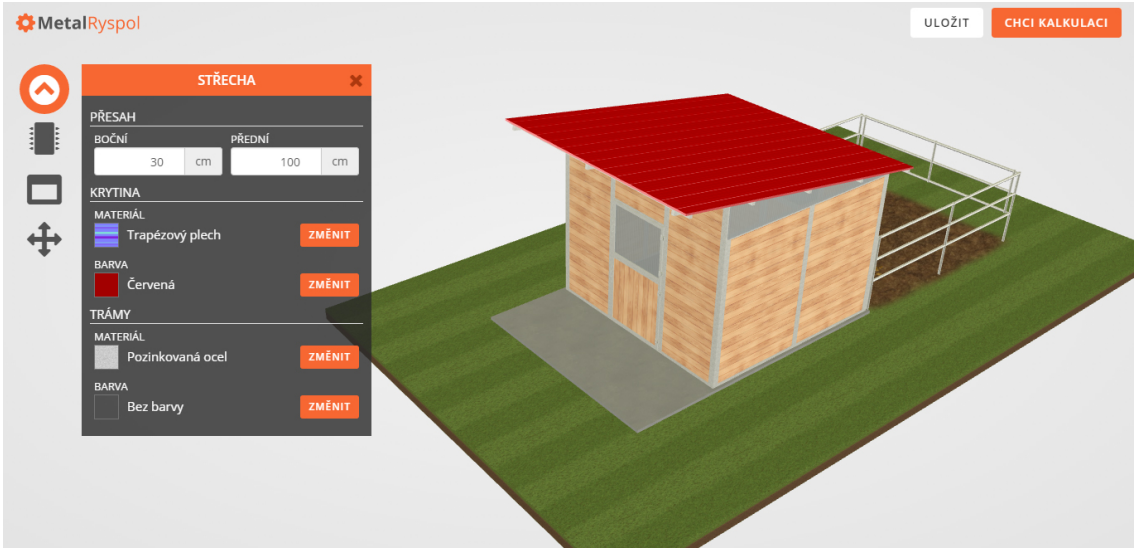
Hlavním úkolem webu je vyvolávat u návštěvníků zájem a zvědavost za účelem provedení nějaké akce směrem k firmě – t.j. podat žádost o kalkulaci, poslat dotaz, zavolat apod. Právě podání žádosti o kalkulaci je jedním z nejčastějších dotazů, na které firma musí reagovat. Proto byl vytvořen formulář, který slouží speciálně k tomuto účelu a navíc umožňuje návštěvníkům podat žádost v nějaké standardizované formě.

Formulář je obdobou toho, který používají zaměstnanci v administraci pro ruční vytváření kalkulací. Rozdíl je v tom, že formulář má dvě podoby – základní a rozšířenou. Důvodem je nechvalně známá neochota lidí vyplňovat dlouhé formuláře (Ilinčev, 2016). Aby návštěvník nebyl odrazen je mu nabídnuto k vyplnění pouze to nejdůležitější minimum a pokud má zájem, má možnost zobrazit si rozšířený formulář.

Podání žádosti přes tento formulář přináší výhodu také v tom, že žádost je ihned zanesena do systému a firma tak může, pomocí již dříve zmíněného kalkulátoru, prakticky v řádech minut poskytnout odpověď v podobě cenové nabídky, ke které může být připojena také 3D vizualizace poptávaných boxů (v případě, že jsou poptávány venkovní boxy).

Konfigurátor boxů

O konfigurátoru boxů toho bylo v této práci napsáno již mnoho, pro úplnost uvedeme ještě realizaci uživatelského rozhraní, které slouží pro jeho ovládání. Stejně jako u ostatních částí systému bylo dbáno i zde hlavně na použitelnost. Konečná podoba konfigurátoru je uvedena na níže uvedeném obrázku.



Obrázek 23: Finální podoba konfigurátoru boxů

Konfigurátor zabírá celou plochu displeje, nejsou zde žádné lišty, panely ani nic podobného. Veškeré ovládací prvky včetně hlavního menu jsou zobrazeny přes okno konfigurátoru a jsou navrženy tak, aby co nejméně překrývali vizualizaci. Jednotlivé nástroje jsou dále členěny do tématických skupinek, tak aby nedocházelo k tomu, že by se uživatelé ztráceli ve změti různých tlačítek a jednotlivé nástroje se snáze vyhledávali.

Pokud se uživatel rozhodne uložit si vytvořený model boxů, bude požádán pomocí modálního okna o zadání e-mailové adresy. Na tuto adresu mu poté bude doručena potvrzovací zpráva s odkazy pro editaci a zobrazení uloženého modelu. Uživatel má také možnost požádat o sestavení kalkulace k vytvořenému modelu, v takovém případě již bude vyžadováno vyplnění také jména a telefonního čísla včetně možnosti napsat poznámku.

Reference

Poslední představenou součástí nově vytvořeného webu je přehled referencí. Reference jsou pro firmu důležitým nástrojem k získání důvěry nových zákazníků. Firma má navíc řadu zahraničních referencí, což pro ni představuje zároveň také příležitost pochlubit se.

Za dobu své existence firma zhotovila řadu zakázek po celé České republice, dále také v Německu, Rakousku, na Slovensku a dokonce také v Itálii. Tento fakt může v očích spousty potenciálních zákazníků vyvolat dobrý dojem a proto bylo navrženo řešení v podobě mapy, která tuto širokou působnost firmy vyzdvihne.

Jednotlivé reference jsou na mapě znázorněny pomocí „špendlíku“. po kliknutí na některý ze špendlíků se zobrazí modální okno s detailem reference, kde se nachází popis reference a také její fotogalerie.

7 Závěr

V této práci byl navrhnut a implementován nový systém pro firmu Metal Ryspol, s. r. o. Závěrem bych chtěl zmínit, že se podařilo splnit všechny požadavky kladené na začátku vývoje. Je zřejmé, že pro firmu může být systém velmi užitečný, jelikož předchozí řešení bylo zcela zastaralé a nedostačující současným požadavkům.

Systém neposkytuje firmě pouhý nástroj pro správu obsahu firemního webu, ale také nástroj pro výrazně rychlejší komunikaci se zákazníkem při výpočtu cenových nabídek. Zároveň přináší originální systém vizualizací produktů firmy, který můžeme považovat za pomyslnou třešničku na dortu. Výsledné řešení je v současné době ještě nutné řádně otestovat a naplnit reálnými daty, aby mohlo být nasazeno do ostrého provozu. Dočasně se však nachází na adrese `http://www.metalryspol.cz/new`.

Tato práce přinesla mnoho užitečného také pro mne samotného. Musel jsem řešit některé problémy, se kterými jsem se nikdy před tím nesetkal a v tuto chvíli, kdy píšu toto zhodnocení, sám na sobě cítím, kolik jsem se toho naučil za tak krátkou dobu. Proto i přesto, že jsem jejímu vypracování musel obětovat spoustu svého času, jsem za tuto příležitost vděčný.

7.1 Navrhovaná vylepšení

Uvedením systému do provozu, práce na jeho vývojí nekončí. Naopak je potřeba jej dále vyvíjet a udržovat, aby nedošlo k jeho brzkému zastarání. Již před začátkem vývoje se počítalo s tím, že některé jeho součásti budou v budoucnu ještě rozšířeny či upraveny.

Velký prostor pro vylepšení se nabízí zejména u konfigurátoru boxů. Ten je v současné době v podobě, kdy poskytuje základní ovšem dostačující možnosti konfigurace. Firma však nejprve chce otestovat, jak se osvědčí a pokud ano, ráda by jej ještě rozšířila o další funkce a možnosti.

Dále vidím možnost vylepšení systému v implementaci notifikací například formou e-mailu nebo SMS zprávy, které by upozorňovali například na novou žádost o kalkulaci, tak aby firma nemusela neustále sledovat systém.

8 Reference

The PHP Group. *PHP: History of PHP - Manual*. [online]. 2016 [cit. 2016-11]. Dostupné z: http://php.net/manual/en/history.php.php.

W3Techs. *Usage of server-side programming languages for websites*. [online]. 2016 [cit. 2016-11]. Dostupné z: https://w3techs.com/technologies/overview/programming_language/all.

Stoupa, Václav. *Přehled a vývoj PHP frameworků*. [online]. 2008 [cit. 2016-11]. Dostupné z: https://www.root.cz/clanky/prehled-a-vyvoj-php-frameworku/.

Hanel, David. *Terminologický slovník*. [online]. 2009 [cit. 2016-11]. Dostupné z: https://java.vse.cz/jsf/chunks/go01.html.

Pitt, Chris. *Pro PHP MVC*. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2012. Expert's voice in open source. ISBN 978-143-0241-652.

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

Monus, Anna. *10 PHP Frameworks For Developers – Best Of*. [online]. 2015 [cit. 2016-11]. Dostupné z: http://www.hongkiat.com/blog/best-php-frameworks/.

Surguy, Maks. *History of Laravel PHP framework, Eloquence emerging*. [online]. 2013 [cit. 2016-11]. Dostupné z: https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/.

Otwell, Taylor. *Laravel – The PHP Framework For Web Artisans*. [online]. 2016 [cit. 2016-11]. Dostupné z: https://laravel.com/.

Zaninotto, Francois. *Two years of symfony*. [online]. 2007 [cit. 2016-11]. Dostupné z: http://symfony.com/blog/two-years-of-symfony.

Koutný, Jiří. *Symfony2, těší mě!*. [online]. 2013 [cit. 2016-11]. Dostupné z: https://www.zdrojak.cz/clanky/symfony2-tesi-me/.

Nette Foundation. *About the Foundation*. [online]. 2016 [cit. 2016-11]. Dostupné z: https://nettefoundation.com/.

Nette Foundation. *Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework*. [online]. 2016 [cit. 2016-11]. Dostupné z: https://nette.org/cs/.

Yii Software LLC. *About Yii - Introduction - The Definitive Guide to Yii 2.0*. [online]. 2016 [cit. 2016-11]. Dostupné z: http://www.yiiframework.com/doc-2.0/guide-intro-yii.html.

British Columbia Institute of Technology. *CodeIgniter Web Framework*. [online]. 2016 [cit. 2016-11]. Dostupné z: https://www.codeigniter.com/.

Šubrta, Václav. *Návrh uživatelského rozhraní webové aplikace*. [online]. 2014 [cit. 2016-11]. Dostupné z: http://gml.vse.cz/data/oppa-webdesign/ui.html.

Otto, Mark. *Bootstrap in A List Apart No. 342*. [online]. 2012 [cit. 2016-11]. Dostupné z: http://markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342/.

ZURB, Inc. *Foundation | About Foundation*. [online]. 2016 [cit. 2016-11]. Dostupné z: http://foundation.zurb.com/showcase/about.html.

Klepáč, Vladimír. *Koní v Česku tryskem přibývá, nejvíc koňáků je v Praze*. [online]. 2016 [cit. 2016-12]. Dostupné z: https://www.novinky.cz/domaci/404363-koni-v-cesku-tryskem-pribyva-nejvic-konaku-je-v-praze.html.

Čápka, David. *UML - Use Case Diagram*. [online]. 2013 [cit. 2016-12]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/uml/uml-use-case-diagram-pribyva-nejvic-konaku-je-v-praze.html.

Arlow, Jim. *UML a unifikovaný proces vývoje aplikací : Objektově orientovaná analýza a návrh prakticky*. Brno: Computer Press, 2008. 387 s. ISBN 80-7226-947-X..

Wikipedia. *Sekvenční diagram*. [online]. 2016 [cit. 2016-12]. Dostupné z: https://cs.wikipedia.org/wiki/Sekvenční_diagram.

Can I use. *WebGL - 3D Canvas graphics*. [online]. 2016 [cit. 2016-12]. Dostupné z: http://caniuse.com/search=webgl.

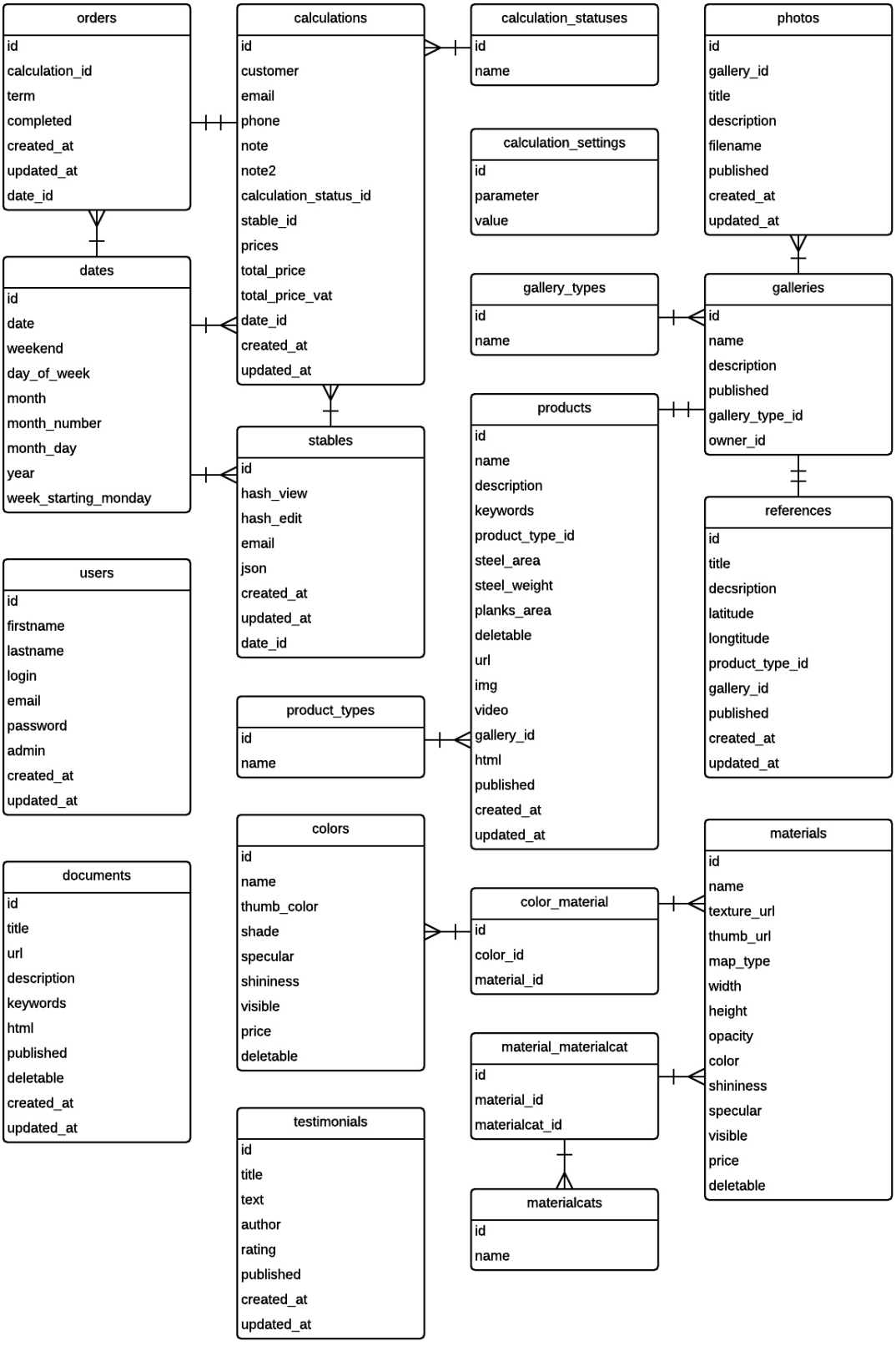
PixelPerfect. *Grafický návrh webu*. [online]. 2015 [cit. 2016-12]. Dostupné z: http://designwebu.com/graficky-navrh-webu.

Wikipedia. *Constructive solid geometry*. [online]. 2016 [cit. 2016-12]. Dostupné z: https://cs.wikipedia.org/wiki/Constructive_solid_geometry.

Ilinčev, Ondřej. *18 nejhorších chyb webových formulářů*. [online]. 2016 [cit. 2016-12]. Dostupné z: http://www.ilincev.com/chyby-formularu.

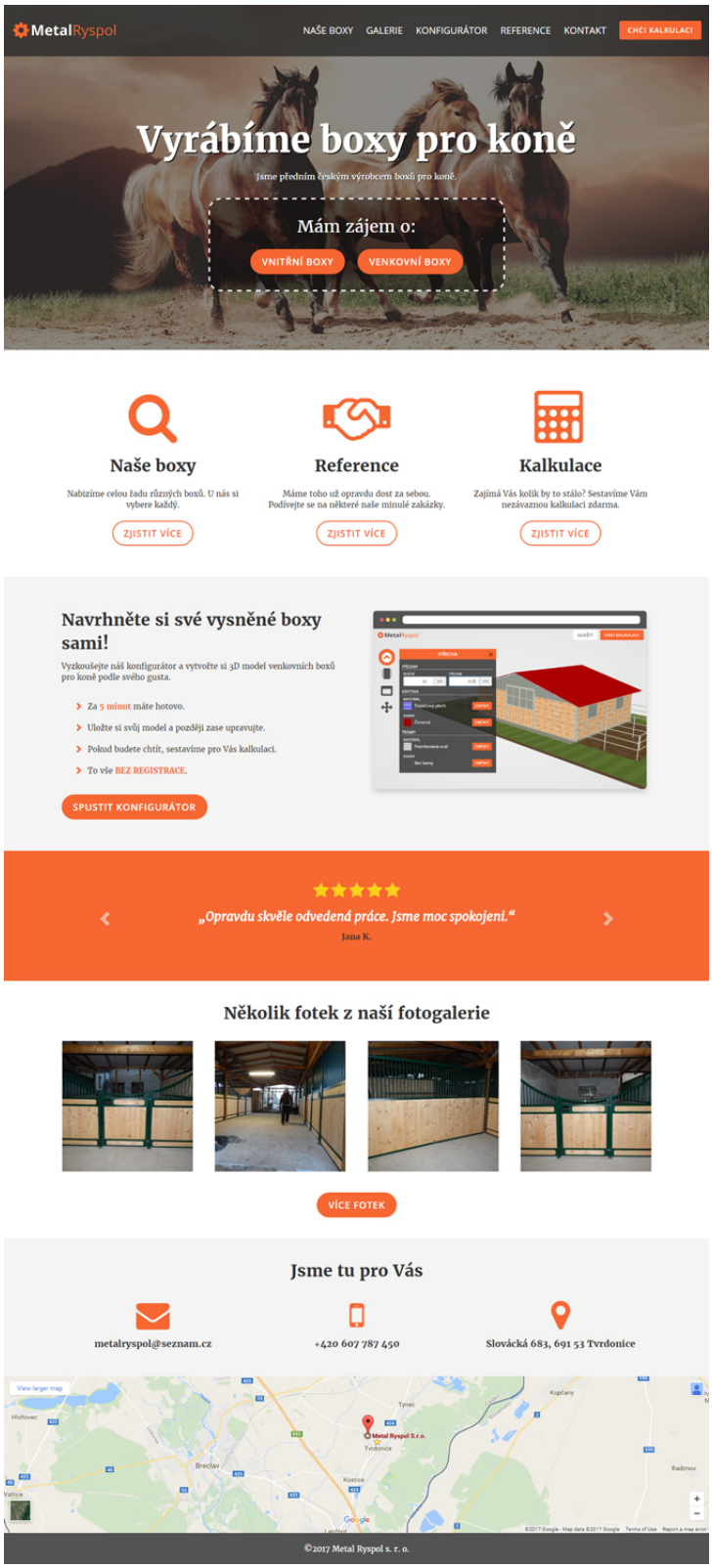
Přílohy

A Schéma databáze



Obrázek 24: Schéma databáze

B Úvodní strana nového webu



Obrázek 25: Úvodní strana nového webu