Mendelova univerzita v Brně
Provozně ekonomická fakulta

Objednávkový systém a webová prezentace firmy Back2Life

Bakalářská práce

Vedoucí práce:
Ing. Dita Dlabolová

Lukáš Augusta

Brno 2015

Tímto bych chtěl poděkovat paní Ing. Ditě Dlabolové za odborné konzultace a vedení bakalářské práce. Poděkování patří i firmě Justmighty s.r.o., která mi zprostředkovala zadání práce. Také děkuji pánům Tomáš Pohl a Gianmarco Salatin za grafické konzultace a odborné postřehy a připomínky.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Objednávkový systém a webová prezentace firmy Back2Life**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

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

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

V Brně dne 2. května 2015 _____

Abstract

Augusta, L. Order management system and website of Back2life business. Bachelor thesis. Brno: Mendel University, 2015.

The thesis deals with design and development of website and order management system for Back2life company. We can divide it into three parts. The first part is devoted to web applications and web development, object oriented programming in PHP and Laravel framework. The second part consists of order management system analysis and description which says how the website and the system were implemented. The last part deals with a summary and a discussion about further necessary steps to run the system. A final evaluation of the system is included.

Keywords

Order management system, information system, web application, PHP, web technologies, PHP framework, Laravel, requirements analysis, UML.

Abstrakt

Augusta, L. Objednávkový systém a webová prezentace firmy Back2Life. Bakalářská práce. Brno: Mendelova univerzita v Brně, 2015.

Práce se zabývá návrhem a tvorbou webové prezentace a objednávkového systému pro firmu Back2life. Můžeme ji rozdělit do tří částí. První část je věnována webovým aplikacím a jejich vývoji, objektově orientovanému programování v jazyce PHP a frameworku Laravel. Druhou část tvoří analýza objednávkového systému a popis, jak byl systém a web implementován. Poslední část se zabývá shrnutím a diskuzí dalších potřebných kroků ke spuštění systému a jeho celkovým zhodnocením.

Klíčová slova

Objednávkový systém, informační systém, webová aplikace, PHP, webové technologie, PHP framework, Laravel, analýza požadavků, UML.

Obsah

1 Úvod

V dnešní době se mnoho lidí zajímá o zdravý životní styl. Zajímají se, jak správně cvičit a věnují větší pozornost potravinám a jídlům, které konzumují.

To otevřelo spoustě zaběhnutým a novým firmám prostor na trhu. Nabízí stále větší množství produktů a služeb týkajících se této problematiky. Avšak pouze zlomku se podaří prosadit na trhu, zaujmout a dobře vést celkový proces prodeje. Spojí-li se více faktorů dohromady, nastává velká šance úspěchu.

Mezi dnešní silné nástroje podporující prodej patří internetový marketing, s nímž souvisí kvalitní web a případně webová aplikace, která nabízí uživatelům jistý komfort například při objednávání produktů nebo při zefektivnění chodu firmy.

Proto jsem se rozhodl v rámci této bakalářské práce poskytnout pomoc firmě Back2life, která má na rozdíl od svých konkurentů dobře nachystanou svoji podnikovou strategii a vyznačuje se značnou kvalitou produktů a služeb.

2 Cíl práce

Po dohodě s firmou Back2life jsme se rozhodli, že vytvoříme pro firmu kompletní webové zázemí. Firmě stále chybí vhodná webová prezentace, aby mohla nabízet své služby klientům, ale zároveň již firma určitou dobu funguje a bylo by dobré, kdyby se zefektivnil chod firmy v podobě automatizovaného objednávkového systému.

Cílem mojí práce je vhodně navrhnout webovou prezentaci, na které zákazníci naleznou všechny potřebné informace a která začne společně s internetovým marketingem prodávat.

Hlavní částí mého úkolu se však stává implementace zmíněného objednávkového systému, díky němuž si klienti budou moci objednávat produkty a služby firmy. Snahou bude nasadit systém do produkčního režimu a otestovat jej v praxi.

Veškerá práce na projektu zahrnuje i grafické řešení a jak webová prezentace, tak i objednávkový systém bude uzpůsoben všem dostupným uživatelským zařízením. Responzivní aplikaci mohou totiž uživatelé otevřít kdykoliv a na jakémkoliv zařízení, čímž se zvyšují šance prodeje.

3 Vývoj webových aplikací

S rozvojem webových technologií vzniká stále více aplikací. V době, kdy svět pohlcuje internet, je to logické. Spousta lidí by si nedokázalo představit situaci, kdy by si nemohli nakoupit nebo si zarezervovat nějakou službu či produkt pomocí internetu. Dnes je internetový obchod často hlavním pilířem prodeje každého podnikatele.

3.1 Webová aplikace

Webová aplikace je taková aplikace, která využívá webový prohlížeč jako klienta. Jde o dynamické webové stránky, kde na straně serveru je potřeba naprogramovat určitou funkční logiku, jako je interakce s uživatelem, připojení k databázi nebo generování datových výstupů pro prohlížeče. Příkladem webových aplikací mohou být online bankovnictví, sociální sítě, rezervační systémy, elektronické obchody, interaktivní hry, online fóra, systémy pro správu obsahu a mnoho dalších. [2]

3.2 Životní cyklus vývoje webové aplikace

Existují různé přístupy, kterými se liší vývoj softwaru nebo aplikací. Tyto přístupy bývají často nazývány modely vývojového procesu softwaru. Mezi známé modely patří například model vodopádu, inkrementační model, V-model, iterativní model atd. Každý model dodržuje svůj životní cyklus v takovém pořadí, aby bylo dosáhnuto požadovaných výsledků.

Životní cyklus aplikace popisuje jednotlivé fáze vývoje. Každá fáze životního cyklu aplikace musí vyprodukovat určité výsledky, které jsou žádoucím vstupem pro další fázi. Veškeré požadavky aplikace jsou převedeny do jejího návrhu, od kterého se dále odvíjí implementace. Této fázi se říká fáze realizační. Po implementaci aplikace přichází na řadu testování, pomocí kterého ověřujeme, zda aplikace obstála a zda je dosáhnuto očekávaných výsledků vzhledem k původním požadavkům.

V každém životním cyklu vývojové aplikace se vyskytuje těchto šest fází:

- sběr požadavků a analýza;
- návrh;
- implementace;
- testování;
- zprovoznění;
- údržba.

3.2.1 Sběr požadavků a analýza

V této fázi jsou shromažďovány veškeré obchodní požadavky, které by měla aplikace splňovat. V období analýzy vystupují především projektoví manažeři a zainte-

resované osoby. Na setkáních manažerů, zainteresovaných osob a uživatelů se řeší otázky typu:

- Kdo bude aplikaci využívat?
- Jakým způsobem se bude aplikace využívat?
- Jaká data a jakým způsobem budou do aplikace vstupovat?

Snahou je, aby tyto obecné otázky byly co v největší míře vyřešeny. Jakmile je sběr požadavků dokončen, přichází na řadu jejich analýza, zda vyhovují a mohou být začleněny do dalších procesů. Výsledkem se stává dokument se specifikací požadavků.

3.2.2 Návrh

V této fázi je připravován návrh aplikace ze vzniklé specifikace požadavků, která byla získána v prvotní fázi. Návrh systému pomáhá upřesnit hardwarové a softwarové požadavky a definuje celkovou systémovou architekturu.

3.2.3 Implementace

Po zpracování veškerých návrhových dokumentů přichází na řadu fáze implementace. Práce bývá obvykle rozdělena do více částí (moduly, jednotky), na kterých pracuje tým vývojářů. Ti jsou hlavním středem zájmu. Fáze implementace zabírá nejdelší období z celého vývojového procesu.

3.2.4 Testování a zprovoznění

Jakmile skončí vývoj aplikace, je důležité otestovat její funkčnost a zda dokáže opravdu to, co bylo definováno v počátečních požadavcích. Po úspěšném testování aplikace a rozhodnutí o jejím spuštění je předána zákazníkovi a je uvedena do provozu.

3.2.5 Údržba

Je téměř pravidlem, že jakmile začne zákazník používat hotovou aplikaci, čas od času vyplují na povrch další skryté problémy, které je potřeba vyřešit. Podstatou údržbové fáze je péče o aplikaci a doplňování potřebných funkcionalit. [1]

3.3 Používané technologie

K vývoji webových aplikací patří řada nepostradatelných technologií. Tou první asi nejznámější je HTML (HyperText Markup Language). Jde o značkovací jazyk, který slouží k popisu obsahu a struktury webových stránek. [3] Na HTML navazují kaskádové styly (Cascading Styles Sheets). CSS tvoří způsob, jak stylovat a prezentovat stránky. Oproti HTML, které slouží k vyjádření struktury obsahu, CSS řeší jeho prezentaci.

Trojici základních technologií doplňuje JavaScript. JavaScript je skriptovací programovací jazyk používaný na straně klienta, který je navržený primárně pro přidávání interaktivity do webových stránek. [4] V dnešní době se často využívá přídavné knihovny jQuery. Jde o malou, rychlou a elegantní JavaScriptovou knihovnu, která pomáhá a ulehčuje práci s JavaScriptem. Díky ní můžeme například procházet a manipulovat s HTML strukturou, řídit různé události, vytvářet animace nebo lépe ovládat AJAX. [5]

AJAX (Asynchronous JavaScript and XML) je nová technika pro vytváření lepších, rychlejších a více interaktivních webových stránek s pomocí XML, HTML, CSS a JavaScriptem. Standardní způsob pro přenos informací mezi serverem se provádí pomocí synchronních požadavků. To znamená, že pokud vyplníme například formulář, potvrdíme jej, tak jsme přesměrováni na novou stránku, která nám nabídne poskytnuté informace ze serveru. Díky Ajax technice jsme schopni zobrazovat informace ze serveru na aktuální stránce bez potřeby stránku aktualizovat. [6]

3.4 Objektově orientované programování a jazyk PHP

Jazyk PHP je jedním z nejpopulárnějších skriptovacích jazyků, který je používaný již řadu let. Každý měsíc je vyvíjeno milion webových stránek a aplikací napsané právě v jazyce PHP. Cesta tohoto jazyku započala jako vytvoření náhrady za jazyk Perl a postupem času se stával čím dál více populárnější a mocnější (Hayder, 2007).

Hayder (2007) uvádí, že na počátku vývoje PHP neobsahovalo možnosti objektově orientovaného návrhu. První náznaky základních prvků objektového návrhu se objevily až poté, kdy Zeev, Rasmus a Andy přepsali jádro jazyka a uvedli PHP3. Následovala verze PHP4, ale kompletní objektový model se objevil až s uvedením opět přepsaného jádra jazyka, a to v PHP5.

3.4.1 Výhody objektově orientovaného programování

Užitím objektově orientovaného programování (OOP) se vývojářům značně zjednodušuje práce. Mohou si jejich práci rozdělit na menší problémy, které jdou mnohem jednodušeji vyřešit. Hlavními výhodami objektově orientovaného programování jsou:

- **Znovu použitelnost**

  Objekt je entita s danými vlastnostmi a metodami, které komunikují s jinými objekty. Objekt je obvykle vyvíjen k řešení specifické množině problémů. Pokud danou množinu problémů řeší jiní vývojáři, stačí pouze vzít již hotovou třídu objektů a zakomponovat ji do existujícího prostředí. Díky tomu se vyhneme opakované tvorbě kódu známé pod zkratkou DRY (Don't Repeat Yourself).

- **Refactoring**

  Když je potřeba upravovat již hotový projekt, OOP nabízí maximální výhodu, protože objekty jsou malé entity a obsahují vlastnosti a metody, které jsou jejich součástmi. Úprava je tedy jednodušší.

- **Rozšiřitelnost**

  Pokud je potřeba do projektu přidat další funkcionalitu, není nic jednoduššího. Stačí pouze přidat atributy či metody do třídy objektu nebo vytvořit zcela novou třídu odvozenou od jejího rodiče (dědičnost).

- **Efektivnost**

  Použitím objektového návrhu je zajištěna lepší přehlednost a efektivnost kódu než v procedurálním programování. Jelikož dělíme počáteční problém na množinu menších problémů, vývojáři mohou pracovat každý na jiném, a to ušetří mnoho času. Pokud jsou vyřešeny všechny malé problémy, automaticky se vyřeší hlavní komplexní problém (Hayder 2007).

3.5 Architektura MVC

Myšlenka architektury MVC (Model-View-Control) se objevila již v 70. letech, jejímž základem bylo zachování prezentace dat oddělených od veškerých metod a jejich interakcí s daty. Odděluje tedy aplikační logiku od zbytku uživatelského rozhraní. Dobře vyvinutý MVC systém pak umožňuje vývojářům soustředit se pouze na určitou část, aniž by se museli zabývat ostatními.

Architektura MVC se používá především ve webových aplikacích, kde je kladen důraz na oddělení otázek a problémů, s čímž je nepřímo spojen i benefit znovu použití kódu. Vývojáři pracují na jednotlivých modulech, což jim umožňuje rychle aktualizovat, přidávat nebo dokonce i odebírat dané funkcionality. [8]

- **Model**

  Model reprezentuje jádro aplikace. Je částí aplikace, která řídí logiku pracující s daty. Často modely přijímají data z databáze nebo je tam naopak ukládají.

- **View (pohled)**

  Úkolem pohledu je zobrazovat data. Je to ta část aplikace, díky které jsme schopni uživatelovi prezentovat daný výstup.

- **Controller (kontrolér)**

  Kontrolér je část aplikace, která řídí uživatelské interakce. Typicky čte uživatelské vstupy a odesílá vstupy patřičnému modelu. [9]

3.6 Framework pro webové aplikace

Framework pro webové aplikace je jakýsi základ, který je specificky navrhnutý tak, aby usnadnil vývojářům budování jejich aplikace. Tyto frameworky typicky poskytují funkční jádro, které je běžné pro většinu webových aplikací, jako je řízení uživatelovy relace, perzistence dat, užití šablonových systémů apod. Využíváním vhodného frameworku může vývojář často ušetřit významné množství času.

Každý framework je jiný, ale mnoho z nich poskytuje množství užitečných věcí, díky nimž se vývojář vyhne znovu implementování již jednou vyřešených problémů. Mezi nejčastější rysy patří:

3.6.1 Perzistence dat

Základním rysem všech webových aplikací je jejich potřeba ukládat informace a tvořit stránky založené na dynamickém zobrazování uložených dat. Například pokud se uživatel přihlásí do administrace určitého systému pro řízení obsahu (např. blog), vloží nový článek, kde zadá jeho název a další údaje a poté se uloží. Z těchto dat se poté vygeneruje nová stránka, ke které uživatelé přistupují v rámci svého požadavku. Je tedy dynamicky vytvořena právě na základě těchto dat.

Pracuje se s daty, a aby byla zachována jejich perzistence, nabízejí frameworky různé možnosti, jak toho dosáhnout:

Tvoří důsledné rozhraní pro opakovaný přístup k systémům s uloženými daty. Poskytují například převod dat na objektové struktury (objektově-relační mapování). Vylepšují výkon aplikace pomocí cachování nad datovou vrstvou. Zahrnují kontrolu integrity dat, jako je ověřování vztahů mezi objekty nebo potvrzování, zda jsou ve formulářích vyplněná všechna povinná pole apod.

Zjednodušují práci s SQL dotazy a chrání aplikaci před napadením.

3.6.2 Autentizace a řízení relace

Statické webové stránky typicky pracují s kompletně anonymními uživateli. Webové aplikace však často požadují po uživateli vytvoření svého účtu, aby se pracovalo s jeho informacemi naskrz webem. Musí tedy být vytvořena patřičná logika, která tyto oblasti pokryje. Frameworky většinou nabízejí již hotovou logiku pro registraci účtů, autentizaci, přihlašování nebo obnovu hesel.

3.6.3 Zabezpečení

Někdy jsou webové aplikace stavěny tak, aby si stránky mohli prohlížet pouze autorizovaní uživatelé. Frameworky dokážou automaticky rozeznat, zda je uživatel přihlášen nebo ne a vyžadovat po uživateli, aby se přihlásil před tím, než bude moci pokračovat v prohlížení webu.

Jakmile je uživatel autorizován, framework pro něj ověří další specifická povolení. Tato povolení mohou být řízena různými způsoby. Uživatel může být zkontro-

lován na základě jeho role nebo mohou být použity jiné metody. Ty jsou obvykle pod kontrolou vývojáře webu a jsou zaneseny v kódu aplikace.

3.6.4 Kešování

Pro zlepšení výkonu aplikace vývojáři často kešují určitý obsah, což znamená, že se tento obsah ukládá do mezipaměti, a díky tomu není potřeba ho znovu načítat při opakovaném spuštění stránek.

3.6.5 Šablonovací systémy

Šablony webových stránek pomáhají udržet logiku oddělenou od zobrazovaného obsahu, což je běžně považováno za dobrou praktiku. Framework může poskytovat jeden nebo více šablonovacích systémů. Pomocí nich je vývojář schopen zobrazit na výstup požadovaná data, která obdržel od patřičných kontrolérů. [10]

3.7 Framework Laravel

3.7.1 O frameworku Laravel

Laravel je silný PHP framework, který napsal Taylor Otwell. Jeho první verze vyšla v roce 2011. Jde o zcela nový framework, který si získává stále větší oblibu. Dnes je jedním z nejoblíbenějších a čeká ho slibná budoucnost. [11]

Jde o framework s elegantní syntaxí, jehož cílem je osvobodit vývojáře aplikací od programování často používaných úloh, jako je směrování adres (routing), autentizace, sessions (relace) nebo kešování (caching). Poskytuje silné nástroje, které jsou vhodné k tvorbě velkých robustních aplikací. Je vhodný na projekty všech velikostí. Využívá také již ověřené komponenty ze známého frameworku Symfony, které poskytují solidní základ dobře testovatelného a spolehlivého kódu. Výhodou frameworku je, že se dá snadno pracovat s nástrojem Composer, který v daném projektu řídí veškeré knihovny třetích stran. Důležitou vlastností je také kompatibilita všech známých používaných databází, jako jsou MySQL, PostgreSQL, SQL Server a SQLite. [12]

3.7.2 Composer

Composer je jednou novinkou ze světa PHP. Jde o nástroj pro řízení přídavných prvků napsaných v PHP. Umožňuje manipulovat s veškerými přídavnými moduly aplikace.

Tento nástroj umožňuje deklarovat závislé knihovny pro daný projekt a instalovat je přímo do něj. Funguje tak, že třetí strana vydá přídavný funkcionální balíček a uživatel si poté může stáhnout jeho danou verzi a nainstalovat ji do svého projektu. Composer není úplně balíčkový manažer. Ačkoliv pracuje s balíčky nebo knihovnami, neinstaluje je globálně, ale pracuje pouze v rámci jednoho projektu. [13]

3.7.3 Artisan

Velice důležitým pomocníkem při tvorbě aplikace a psaní kódu je vlastní rozhraní příkazové řádky zvané *Artisan*. Toto rozhraní poskytuje velké množství pomocných příkazů, které když se umí správně použít, dokážou ušetřit mnoho času.

Nabízí tedy spoustu možností, které nám pomohou při tvorbě různých tříd sloužících ke spouštění nejrůznějších událostí. Díky němu dokážeme pomocí jednoduchého příkazu vytvořit připravený model, pohled, kontrolér, migraci, třídy definující požadavky, plnit databázi a tak dále. V dalších případech existuje možnost vytvořit si vlastní příkazy, které se dají opakovaně používat. [14]

3.7.4 Eloquent ORM a Query builder

Většinou je potřeba aplikaci propojit databází. Nástroj zvaný *Eloquent ORM* (Object-relational mapping), který přišel s *Laravelem*, ji neskutečně usnadňuje. Zajišťuje základní operace s databází a poskytuje nastavení mnoha vztahů mezi tabulkami, např. one-to-one, one-to-many, many-to-many ale i další.

Eloquent nám umožňuje v MVC struktuře vytvořit model, který koresponduje s databází. Vezme data z databáze a uloží je do jednotlivých tříd. Konkrétní záznamy náležící dané třídě pak představují jednotlivé objekty. [15]

S pojmem *Eloquent ORM* souvisí i pojem *Query builder*, který poskytuje jednoduché rozhraní k vytváření a spouštění databázových dotazů. Může být využit k vykonání veškerých databázových operací. Zvládá počáteční databázové spojení, vytváření, čtení, mazání nebo aktualizování záznamů v databázi, spojování tabulek apod. Funguje na většině databázových systémů.

Značnou výhodou používání *Query builderu* je, že se nemusíme bát a starat o SQL útoky. Můžeme se vyhnout veškerým řádkům kódu, které právě tento problém ošetřují. [16]

3.7.5 Routing

Jedním ze základních rysů frameworku je tzv. routing. Jde o mapování URL adres k patřičným akcím daných kontrolérů. Umožňuje nám definovat pěkné adresy a dělat je konfigurovatelnými a flexibilními. [17] Mějme například konkrétní zápis pro route:

```
Route::post('obchod/pridat-produkt', 'ProductController@store']);
```

Tento route nám vypovídá o tom, že zadáme-li požadavek na adresu *www.example.cz/obchod/pridat-produkt*, odkážeme se do *kontroléru ProductController* a vyvoláme patřičnou akci k přidání produktu, která se jmenuje *store*.

3.7.6 Autentizace

Framework *Laravel* je napsán tak, aby pokryl praktické a často opakující se problémy, jako je právě autentizace. Autentizace v *Laravelu* je velice dobře připravena a nastavena. Jsou předpřipravené formuláře a pozadí pro registraci a přihlášení,

které se dají lehce přizpůsobit vlastním potřebám. Další vestavěnou funkcí *Laravelu* je možnost obnovení hesla uživatele v případě, že by ho zapomněl. [18]

3.7.7 Middlewares

HTTP *middleware* poskytuje v aplikaci mechanismus, který filtruje HTTP požadavky vstupující do dané aplikace. Říká nám, co se má stát, pokud nastane daná situace. V *Laravelu* je předpřipravených několik *middlewarů*, ale existuje možnost napsat si svůj vlastní. Jedním z nich je takový, který ověřuje uživatele. Pokud se nejedná o ověřeného uživatele, *middleware* ho přesměruje například na stránku s přihlašováním. V opačném případě se pokračuje v procesu požadavku a je mu dovolen další přístup do aplikace. [19]

3.7.8 Migrations a database seeding

Migrace poskytují vývojáři způsob, jak modifikovat schéma databáze a přitom současný stav zachovat. Jednoduše řečeno, díky migracím máme možnost manipulovat s databází ne přes databázové rozhraní, ale můžeme ji měnit pomocí napsaného kódu. [20]

Během vývoje aplikace je potřeba neustále dělat změny, což díky *Laravelu* můžeme provádět velmi efektivně. Můžeme lehce přidávat tabulky, přejmenovávat sloupce, přidávat indexy apod. To nám umožňuje dělat robustní změny na již zaběhnuté aplikaci. Příkladem v naší aplikaci může být vytvoření tabulky *users*:

```
class CreateUsersTable extends Migration {

	/**
	* Run the migrations
	*/
	public function up()
	{
		Schema::create('users', function(Blueprint $table)
		{
			$table->increments('user_id');
			$table->string('first_name', 25);
			$table->string('last_name', 35);
			$table->string('phone', 30);
			$table->string('email')->unique();
			$table->string('password', 60);
			$table->integer('deposit');
			$table->text('note');
			$table->rememberToken();
			$table->timestamps();
		});
	}
	/**
	* Reverse the migrations.
	*/
	public function down()
	{
		Schema::drop('users');
	}

}
```

Tvorba aplikace je pouze jedním krokem k jejímu spuštění. Důležitým dalším krokem při vývoji aplikace je otestovat aplikaci nějakými daty. *Laravel* zahrnuje speciální plnící třídy, které dokážou naplnit databázi fiktivními daty a které poté využijeme k testování aplikace. Tomuto procesu se *říká database seeding*. [21]

3.7.9 Šablonovací systém Blade

V každé MVC aplikaci je důležité, jak oddělit zobrazovaný obsah od logiky aplikace a jak se budou získaná data prezentovat. *Blade* je jednoduchý šablonovací systém, který byl vytvořen pro *Laravel*. Na rozdíl od obsahu, který generuje samotný programovací jazyk PHP, je obsah vygenerovaný pomocí *Blade* daleko čistější a přehlednější. Systém umožňuje provádět tyto procesy:

- definovat sekce;
- rozšiřovat pohledy o jiný obsah;
- vypisovat obsah;
- provádět podmínkové příkazy;
- provádět cykly;
- zahrnovat další obsah. [22]

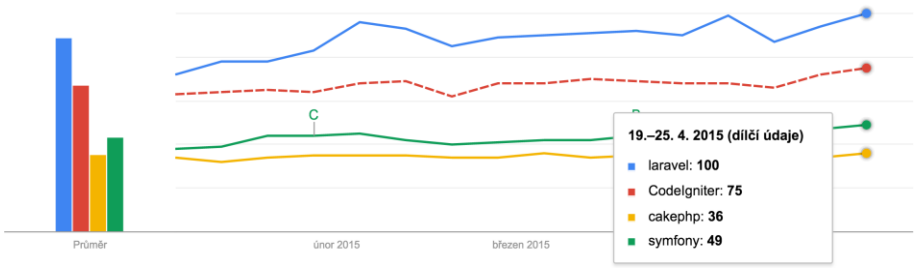
3.7.10 Porovnání s ostatními frameworky

Není jednoduché říci, který framework je nejlepší či nejoblíbenější. Vede se nespočet sporů a diskuzí ohledně tohoto tématu. Každý z těchto frameworků má ale své výhody a nevýhody. Je důležité, pro jaký účel daný framework volíme a v jak velkém rozsahu chceme tvořit webovou aplikaci. Hodnotí se také výkon a rychlost zpracování požadavků.

V roce 2015 je celkově mezi uživateli nejoblíbenější právě *Laravel*. V České republice není příliš rozšířený a spíš stále převládá známý framework *Nette*. Dle mého názoru je tomu tak proto, že jde o český framework s česky psanou dokumentací. Mnoho lidí odradí dokumentace ostatních frameworků, které jsou psány v jazyce anglickém. Není to určitě jediný důvod, protože i *Nette* je kvalitní a ozkoušený framework. Věřím však tomu, že to bude právě *Laravel*, který se nadále udrží na horních příčkách oblíbenosti.

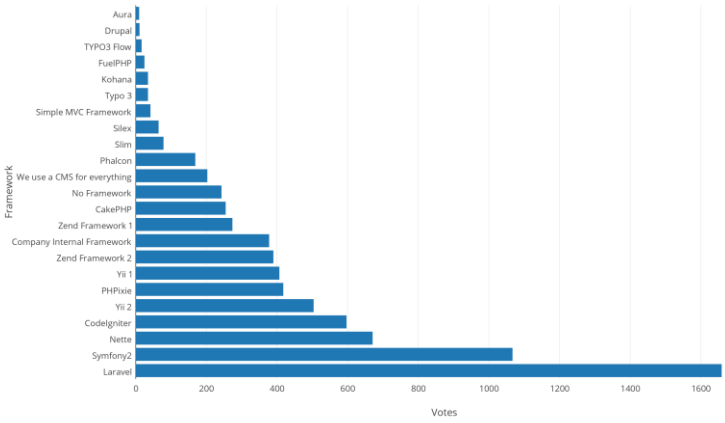
Následující grafy ukazují statistiky používání PHP frameworků:

- Statistika z vyhledávání názvů PHP frameworků získaná z Google Trends - www.google.cz/trends/.

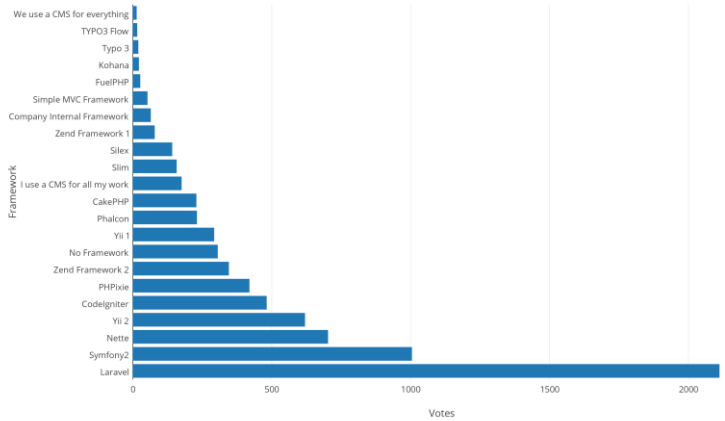


Obr. 1 Graf zájmu o vyhledávání názvů jednotlivých frameworků

- Statistiky nejlepších PHP frameworků pro rok 2015 podle SitePoint - www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/.



Obr. 2 Oblíbenost PHP frameworků ve firmách



Obr. 3 Oblíbenost PHP frameworků v osobních projektech

4 Analýza objednávkového systému

4.1 O firmě Back2life

Firma Back2life se zabývá zdravým životním stylem. Snahou týmu, který ji vede, je poskytnout komplexní službu týkající se zdravého stravování a cvičení. Nejsou tedy jako ostatní konkurenti, protože nabízejí určitý balíček služeb.

První složkou, kterou se může firma charakterizovat je tvorba jídelníčků. Liší se však tím, že jejich jídla jsou nejen ze zdravých surovin, jsou ale také hodnotově vyvážená a pestrá. Suroviny jsou pečlivě vybírány a poté předány do kuchyně, kde se z nich uvaří pro klienty jídla dle připravených jídelníčků. Ta jsou poté pečlivě naservírována do krabiček, které firma další den rozváží svým klientům.

Druhou složku tvoří služba poskytování osobních poradenství a individuálních cvičebních tréninků. Obě složky jsou navzájem nepostradatelné. Klient si domluví schůzku s trenérem, proberou spolu základní informace. Poté se klient s trenérem domluví na své tréninky a paralelně mu je nabídnuta strava podobě krabičkových jídel.

4.2 Požadavky na objednávkový systém

Firma už nějakou dobu funguje, a proto se rozhodla, že by bylo vhodné zefektivnit a zrychlit chod firmy pomocí částečně automatizovaného objednávkového systému, který by dokázal přijímat objednávky jídel svých klientů a novou webovou prezentací, která podpoří prodej.

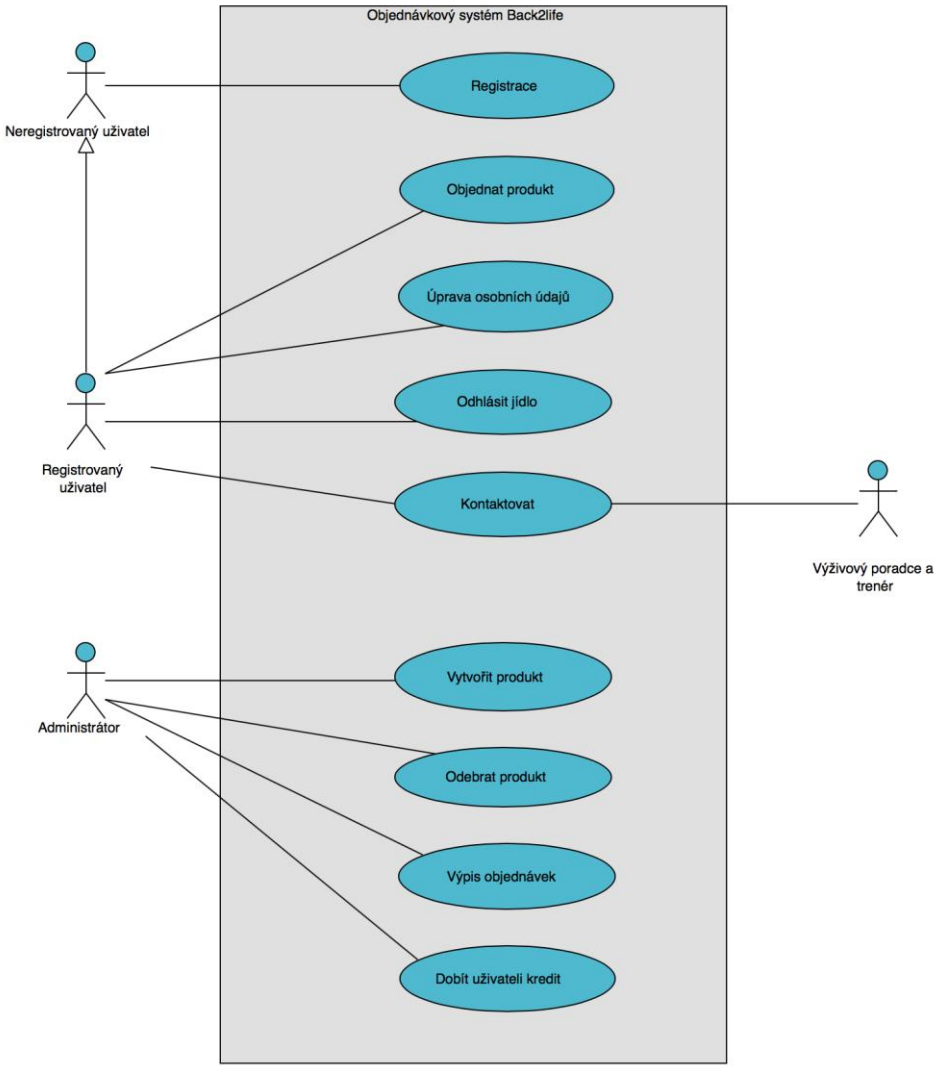
Představa firmy byla taková, aby systém obsahoval dvě strany, a to jednu pro klienty a druhou pro firmu samotnou. Běžný uživatel by měl být schopen registrovat se do systému, vyplnit své osobní údaje, které by mohl v případě potřeby upravit. Jakmile by byl uživatel autorizován, měl by být schopen vybrat si vhodný produkt (krabičky s jídlem) a objednat si ho na určité časové období. Dalším požadavkem firmy bylo, pokud se klient rozmyslí nebo například nebude stíhat sníst určité jídlo v konkrétní den, aby si jej mohl odhlásit. Pokud si odhlašuje jídlo na následující den, může operaci provést pouze do 12:00 dne aktuálního. Také by měl mít možnost kontaktovat pomocí systému svého osobního trenéra.

Na druhu stranu z pohledu firmy je důležité, aby mohla vytvářet a nabízet v systému své produkty (různé typy a formy krabičkových jídel). Tyto produkty poté bude moct volitelně zpřístupňovat pro objednávky klientů. Za nejdůležitější funkcionalitu se však považuje výstup, díky kterému se především kuchaři mohou podívat, kolik a jaká jídla mají na daný den uvařit. V tomto denním výpisu by měli byt schopni seřadit seznam na základě jména klienta, jeho adresy nebo třeba samotného typu produktu. Důležitá pro ně budou čísla, kolik mají například uvařit obědů apod.

4.3 Use case diagram

V návrhu use case diagramu vystupují 4 aktoři. První dva z nich představují běžného uživatele (zákazníka) systému, a to v podobě registrovaného uživatele, který je potomkem uživatele neregistrovaného. Pro přihlášení do systému je uživatel nucen se registrovat. Až poté si může uživatel vybrat produkt, který si chce objednat, nebo si odhlásit vybraná jídla, spravovat své osobní údaje či kontaktovat výživového poradce a trenéra, který do systému vstupuje jako další aktor.

Posledním aktorem je administrátor. Ten jako správce může přidávat a odebírat produkty, vypisovat a filtrovat objednávky zákazníků a dobíjet uživatelům kredit, jakmile zašlou peníze na účet.



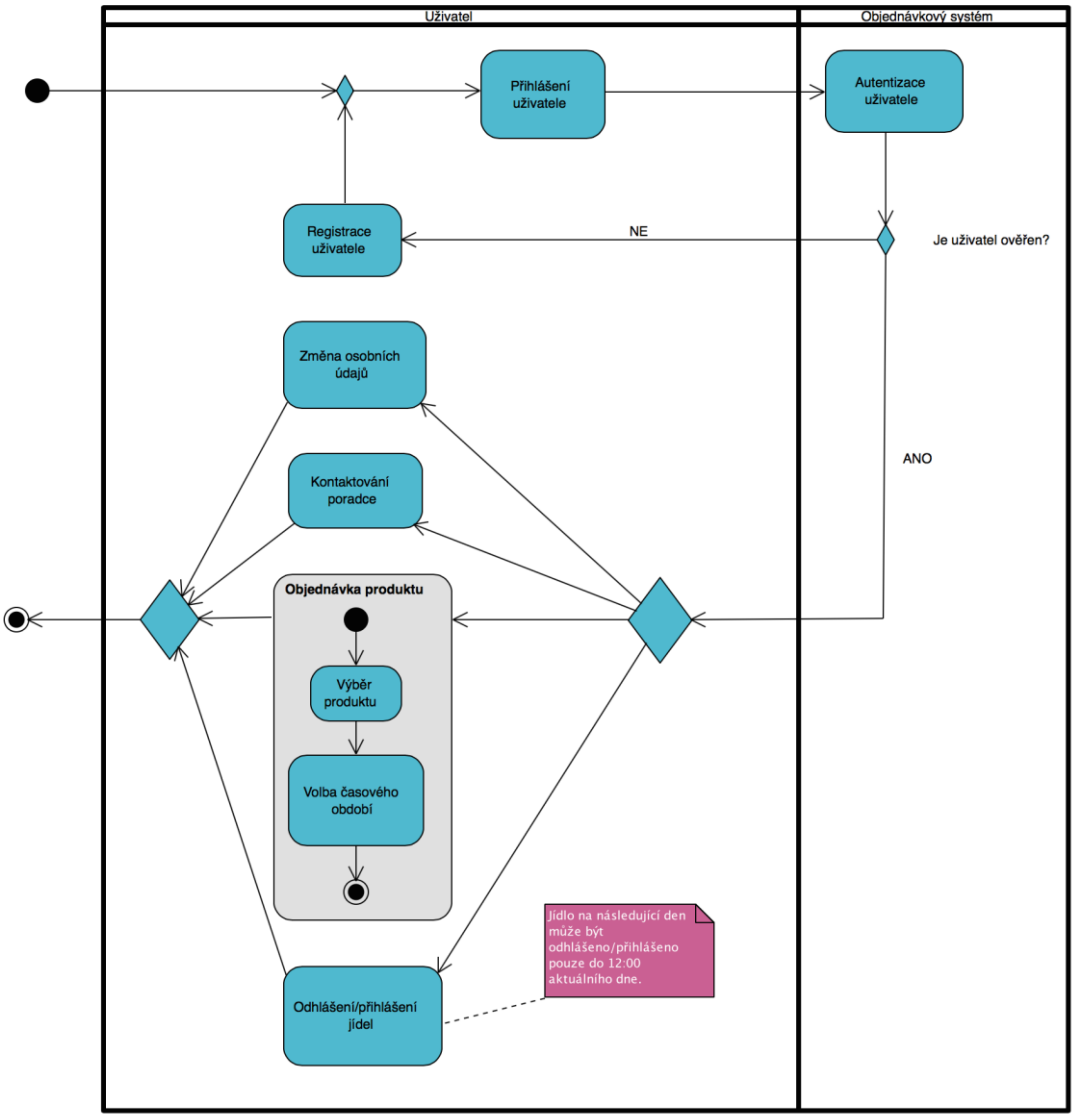
Obr. 4 Use case diagram objednávkového systému Back2life

4.4 Diagram aktivit

Diagram aktivit konkrétního případu představuje průchod systémem ze strany uživatele. Znázorňuje tok aktivit, které na sebe navzájem navazují.

Uživatel začíná tím, že se musí přihlásit do systému. Systém jej poté ověří a na základě výsledku ho pustí dál. V případě neúspěchu ho přesměruje na stránku s registrací. V opačném případě na výběr z několika akcí, které může provést.

Vybere-li si uživatel objednávku produktu, musí si nejprve zvolit, jaký produkt chce a poté si vybrat časové období, na které chce daný produkt objednat. Tento tok akcí je v diagramu znázorněn jako vnořená aktivita.



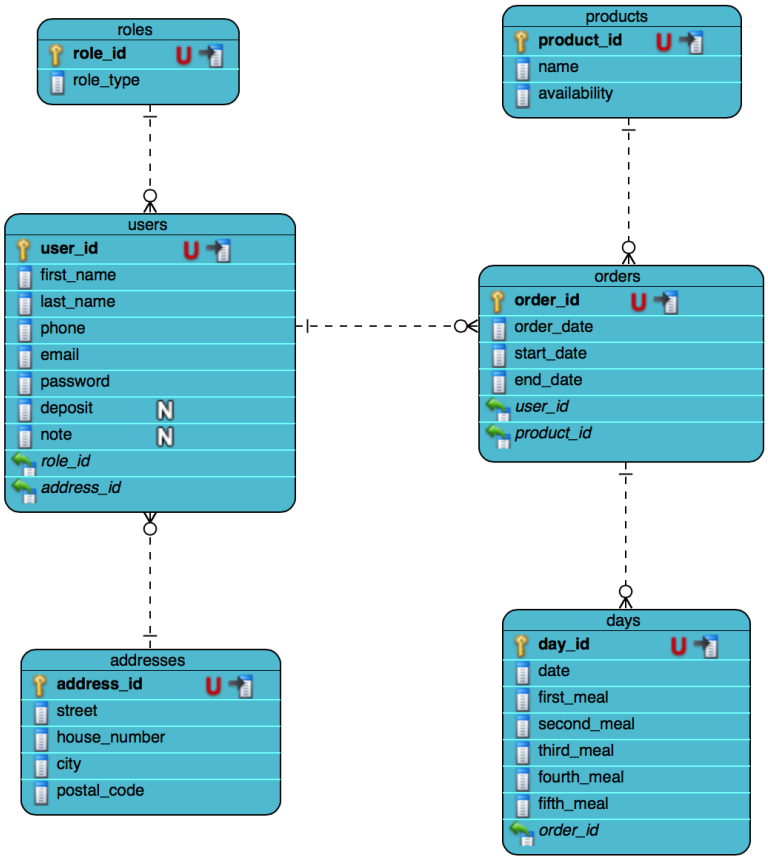
Obr. 5 Diagram aktivit – uživatelův průchod systémem

4.5 Návrh databáze a entitně relační diagram

Na základě požadavků byl pro objednávkový systém vytvořen návrh databáze. Skládá se z šesti tabulek (entit):

- users (uživatelé);
- roles (role);
- addresses (adresy);
- orders (objednávky);
- products (produkty);
- days (dny).

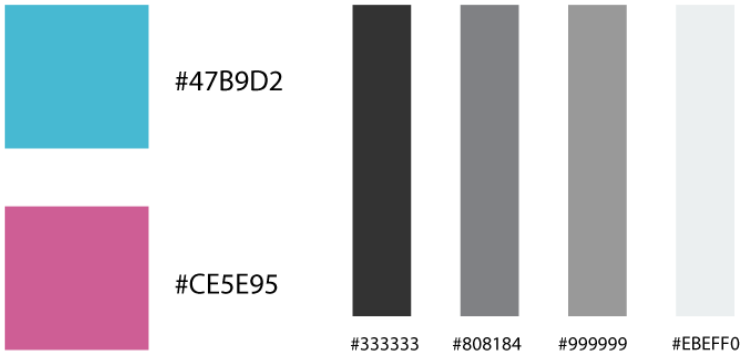
Aby klienti získali přístup do systému, je potřeba je registrovat do databáze včetně jejich osobních údajů a adres. Každý uživatel může zastávat jednu roli, a to buď klasického uživatele (zákazníka) nebo administrátora. Tito uživatelé si objednávají produkty (typy balíčků) na zvolené časové období. Pro každé toto období se zaznamenává jednotlivý den, protože je potřeba manipulovat jednotlivými jídly.



Obr. 6 Entitně relační diagram

4.6 Grafický návrh projektu

Součástí práce bylo také vytvoření patřičného grafického výstupu. Barevné ladění projektu vychází z firemních barev a dodaného loga. Barevnou paletu tvoří dvě základní barvy, které jsou doplněny neutrálními barvami odstínů šedi. Napříč celým projektem jsou používané varianty fontu *Proxima Nova*.



Obr. 7 Paleta barev pro grafický návrh

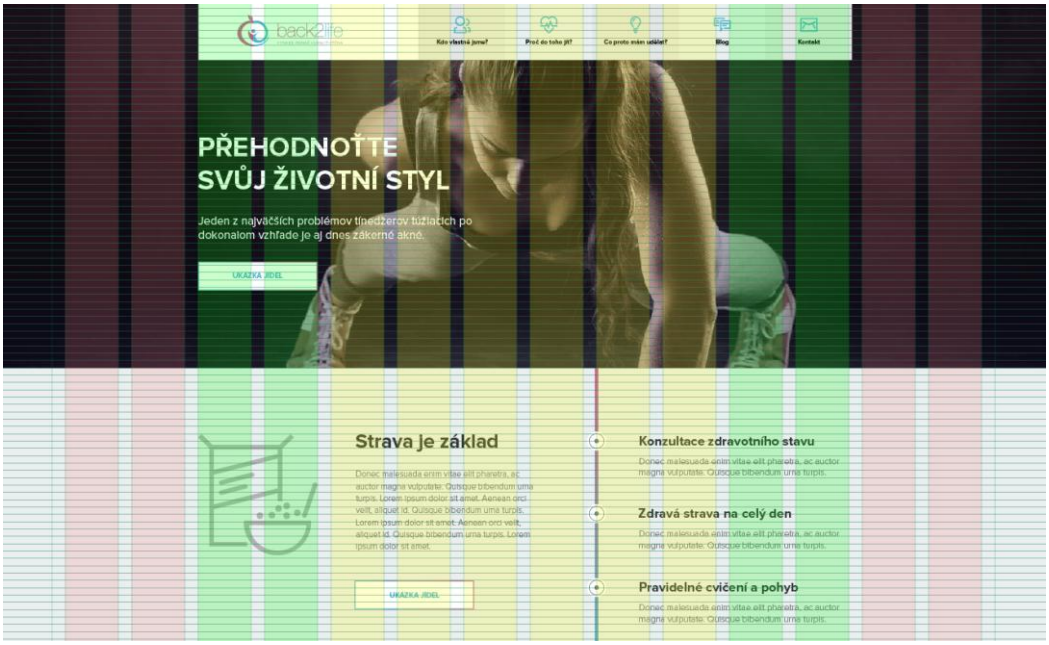
4.7 Návrh webové prezentace

Celá webová prezentace je navržena a zarovnána do mřížky, kterou tvoří 10 sloupců. Těchto deset sloupců určuje maximální možnou šířku zobrazovaného obsahu (1234 px). Zároveň se klade důraz na použití účaří, které je v tomto návrhu vysoké 13 px. Díky pečlivému používání účaří jsme schopni dodržet určitá typografická pravidla, díky nimž se stane text čitelnější, přehlednější a celkově bude opticky působit lépe. Účaří je rámci celého webu používáno i k vertikálnímu zarovnání.

V prvotní fázi půjde o jednostránkový web (onepage). Veškerý obsah se bude odehrávat na jedné dlouhé stránce. Jedinými samostatnými podstránkami budou stránky blogu, do kterého se v rámci marketingu a udržování vtahu s klienty bude pravidelně přidávat obsah. V případě zájmu je zákazníkovi nabídnut odkaz do objednávkového systému.

Kromě hlavičky a zápatí bude web obsahovat hlavní obsah, který bude rozdělen na jednotlivé sekce, ke kterým se odkážeme pomocí navigace:

- úvodní informace a snaha upoutat pozornost;
- informace o lidech, kteří zastupují a pracují ve firmě;
- úryvky z blogu;
- nabídka produktů;
- kontaktní sekce.



Obr. 8 Ukázka z návrhu webové prezentace

5 Implementace

5.1 Webová prezentace

Realizace webové stránky musí přesně odpovídat jejímu návrhu. Nejdříve byla vytvořena *HTML5* struktura a poté byl na ni nasazen vzhled, který zajišťují *CSS*. Pro lepší práci s kaskádovými styly byl použit preprocesor *SASS*, který umožňuje efektivnější zápis, používaní proměnných a mnoho dalších možností.

Snahou při tvorbě *CSS* je dodržet určité metodiky. Jednou z nich je *OOCSS* (Object-oriented CSS). Jde o způsob vytváření kódu tak, aby se stal znovu použitelným na jiném místě a abychom neopakovali kód, což přispěje k jeho lepší správě a i výkonu. [23]

Na tuto metodiku navazuje druhá, která se nachází pod zkratkou *BEM* (Block, Element, Modifier). Jak lze ze zkratky odvodit, vytvoříme hlavní blokový prvek, v němž jsou zanořeny jeho části. Tyto části pak mohou obsahovat modifikující třídy. [24]

```
<ul class=”menu”>
       <li class=“menu__item“>
       <li class=“menu__item“>
       <li class=“menu__item menu__item--active“>
</ul>
```

5.2 Objednávkový systém

Celý systém byl napsán tak, aby pokryl prvotní požadavky zadavatele. Obsahuje veškeré základní potřeby, jako je registrace a manipulace s produkty či objednávkami. Přihlásit se do systému může uživatel v roli zákazníka nebo administrátora.

5.2.1 Registrace a autentizace uživatele

Při prvním navštívení objednávkového systému je po uživateli vyžadovány přihlašovací údaje. Ty získá registrací do systému. Povinnými údaji, které musí vyplnit, jsou:

- jméno;
- příjmení;
- e-mail;
- telefonní číslo;
- a kompletní adresa.

Po úspěšné registraci je uživatel přesměrován na stránku s přihlášením, kde zadá své údaje a na základě správnosti ověření bude uživatel přesměrován dále.

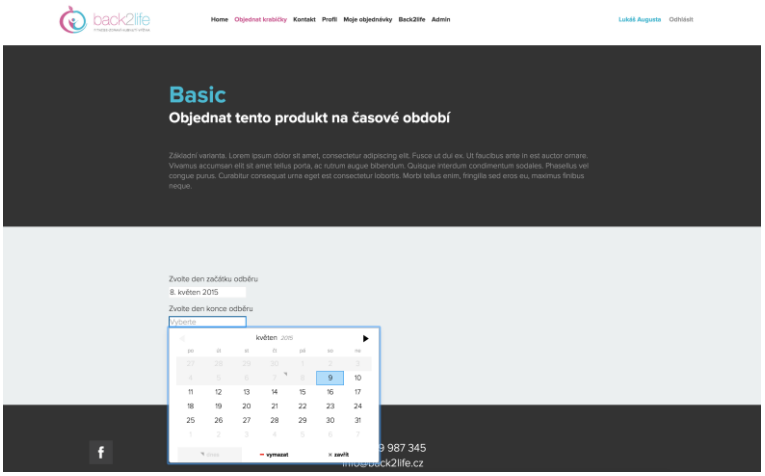
5.2.2 Přidání a odebírání produktu

Je-li přihlášen administrátor, zobrazí se mu v navigaci administrační položka, přes kterou se dostane k volbě přidání či odebrání daného produktu. Chce-li vložit nový produkt do systému, musí jej nejprve pojmenovat a napsat k němu krátký popis, který se bude uživatelům zobrazovat.

Další položkou formuláře je možnost výběru, zda je produkt dostupný či nikoliv. Pokud bude administrátor chtít zveřejnit produkt k objednání až později nebo bude chtít kdykoliv pozastavit jeho prodej, poslouží mu právě tato volba.

5.2.3 Objednávka produktu

Pokud přejde uživatel k objednávce produktu, dostane se na stránku, kde se mu zobrazí dynamický seznam s vytvořenými produkty. Vybere si produkt a dostane se na stránku s výběrem časového období, na které si chce produkt objednat. Datum si vybírá z kalendářů, které jsou implementovány s použitím pluginu *Pickadate.js*.



Obr. 9 Ukázka objednávky produktu

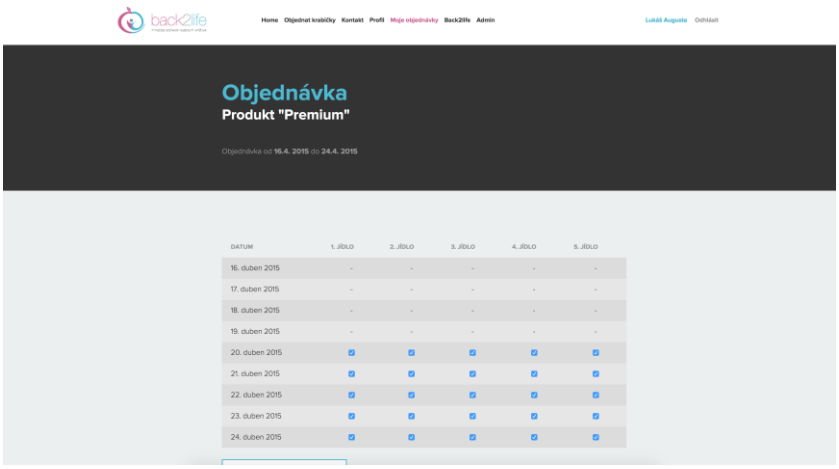
Důležité však bylo nastavit ho tak, aby výběr data odpovídal podnikovým pravidlům, jako je např. objednávka produktu na následující den pouze do 12:00 aktuálního dne.

```
public static function orderCondition($startDate)
{
	$currentDateTime = Carbon::now();
	$date = Carbon::createFromFormat('Y-m-d', $startDate)
		->setTime(0,0,0)->addHours(12);

	if ( $date->subDay(1) > $currentDateTime ) {
		return true;
	} else {
		return false;
	}
}
```

5.2.4 Úprava objednávek

Důležitá je pro zákazníka manipulace s objednanými jídly. Může si svá jídla na dané dny libovolně odhlašovat či přihlašovat. To může podobně jako v předchozím případě učinit pouze do dvanácti hodin aktuálního dne. Je to proto, aby lidé ve firmě lidé stihli nakoupit suroviny a uvařit žádaná jídla.



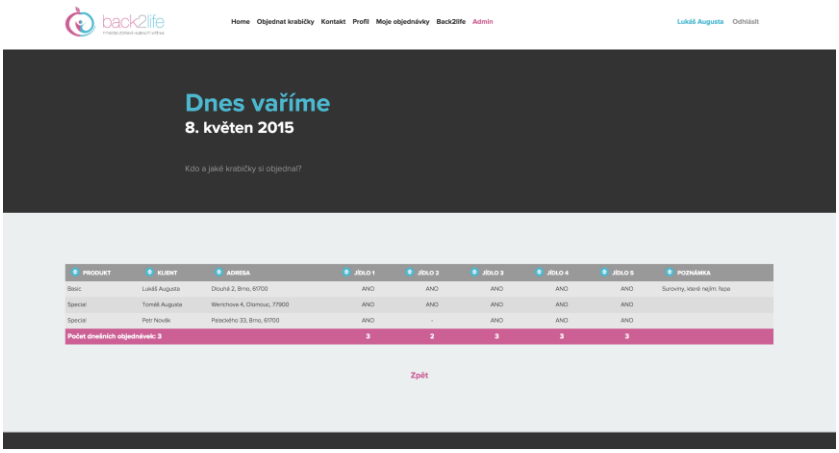
Obr. 10 Ukázka úpravy objednávky

5.2.5 Výpis objednávek

Asi nejdůležitější administrátorovou funkcí je výpis objednávek na určitý den. Díky němuž mají například kuchaři přehled o tom, co mají na daný den uvařit.

Výpis poskytuje informace o tom, kdo a jaký produkt si objednal. Dále obsahuje zákazníkovu adresu, aby se vědělo, kam se mají krabičky s jídlem dovézt. Uživatel má možnost vyplnit ve svém profilu poznámku, do níž může například napsat, které suroviny nemůže konzumovat.

Důležitými údaji jsou záznamy o objednání určitých jídel. Každý uživatel si může totiž například odhlásit oběd nebo večeři apod. Konečná tabulka poskytuje součty objednaných jídel, díky čemuž kuchaři přesně vědí, jaká množství musí uvařit. S tabulkou mohou libovolně pracovat a řadit ji podle jakýkoliv kritérií (např. dle jména uživatele či jeho adresy.)

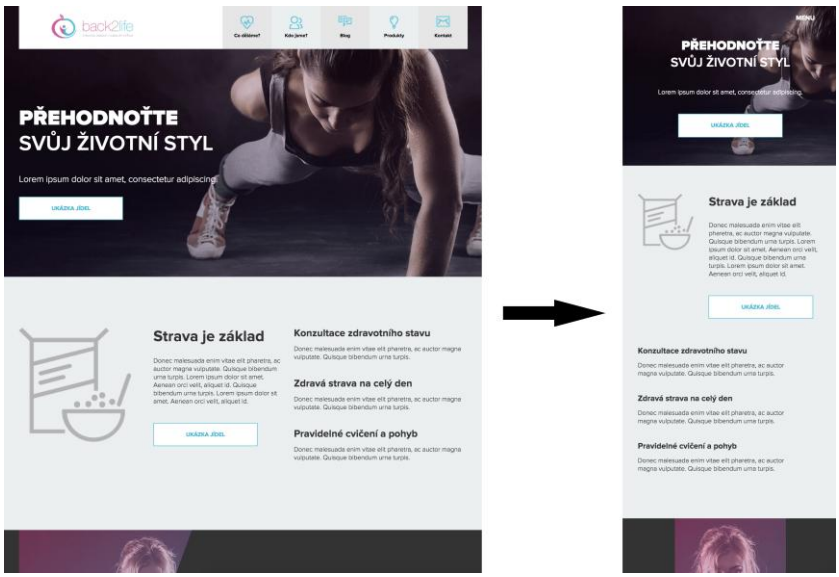


Obr. 11 Ukázka výpisu objednávek

5.3 Responzivní návrh

Jelikož v dnešní době jsou v obrovské míře rozšířena zařízení, jako jsou mobilní telefony nebo tablety, tak je potřeba webový návrh uzpůsobit na míru každému z nich. Stejně tak existuje i řada odlišných velikostí obrazovek.

Proto jsou velikosti veškerých elementů webu, ale i objednávkového systému zaneseny procentuálně, díky čemuž je vzhled předpřipravený změnám velikosti obrazovky. Všude, kde se uspořádání a velikosti prvků začnou rozcházet, jsou použity zlomové body, které slouží k individuálnímu stylování daného prvku.



Obr. 12 Responzivní řešení

6 Diskuze

Základní požadavkům bylo vyhověno a kostra projektu je připravena ke spuštění a testování. Předpokládá se, že první do provozu vkročí webová prezentace, aby se měli zákazníci kam odkazovat a jako druhý bude spuštěn objednávkový systém, do kterého se musí zanést další podněty zadavatele.

6.1 Testování

Webová prezentace a objednávkový systém budou nahrány na hosting pod zkušební doménu, ke které budou mít přístup osoby z firmy Back2life a další lidé, kteří se budou chtít zapojit do testování.

Bude důležité získat zpětnou vazbu a informace, které povedou ke zlepšení systému. Především mě bude zajímat uživatelská zkušenost, a jestli se dá v systému dobře orientovat. Budou nutné vyzkoušet několik scénářů, aby byly odhaleny případné chyby.

Z hlediska výkonu si myslím, že nebude problém, protože *Laravel* funguje velmi rychle a objednávkový systém nebude pod příliš velkým zatížením uživatelů.

6.2 Výběr časového rozmezí objednávky

Na zamyšlenou je také volba časového rozmezí dané objednávky. V základu je objednávkový systém nastaven tak, že jdou zadávat objednávky nejdříve na další den, a to pouze v případě, že je objednávka provedena do dvanácti hodin dne aktuálního. Teprve, jak se získá zpětná vazba a jak usmyslí firma, se rozhodne, od kdy bude možno provést objednávku.

S tím také souvisí nastavení obchodní politiky firmy. Záleží, jestli firma nastaví minimální časové období pro své objednávky nebo umožní svým klientům objednávat jejich služby pouze na určité dny nebo i na kratší období.

6.3 Způsob platby

Dalším tématem je způsob placení služeb a produktů. První variantou je zasílání peněz na bankovní učet. Zákazník zašle peníze na účet firmy a ta po jejich přijetí nabije uživateli v systému kredit, z kterého bude moci čerpat v rámci svých objednávek.

Druhou možnou a dnes moderní variantou je platba přes platební bránu. Uživatel si přímo v systému vybere produkt a rovnou zaplatí. Případně si dobije svůj kredit, ze kterého poté bude čerpat.

Osobně si myslím, že bude vybrána druhá varianta, jelikož jde o metodu rychlejší a nemusí se čekat na převod peněz, ale také mnohem jednodušší, kdy se klient nemusí složitě přihlašovat do svého internetového bankovnictví, ale pouze vyplněním krátkých údajů zaplatí.

Vzhledem k nejasným požadavkům firmy nejsou implementovány jakékoliv funkcionality týkající se práce s kreditem.

6.4 Zhodnocení z ekonomického hlediska

Firma již nějakou dobu funguje. Jídla se však vaří v malých prostorách pro určité množství zákazníků a není zatím takový problém vyřizovat objednávky individuálně. V momentě, kdy se však spustí jejich webová prezentace a bude kladen důraz na optimalizaci pro prohlížeče a internetový marketing, se očekává nárůst zákazníků. V tomto případě již nebude vše tak jednoduché.

Podle výzkumu pro město Brno je po nastartování internetové marketingové kampaně očekávána konverze okolo 100 nových zákazníků za měsíc, která se bude v dalších měsících snižovat a po nasycení trhu se bude udržovat na nižších hodnotách. Návštěvnost webových stránek i systému nebude příliš vysoká, a proto postačí cenově běžný průměrný hosting. Dle růstu firmy a jejích nových požadavků se budou odvíjet náklady na údržbu a rozvoj především objednávkového systému. Předpokládaným nákladem firmy bude rozšíření webové prezentace, kdy každá sekce bude rozšířena o svoji vlastní stránku. Největší položkou pro firmu se však stane internetový marketing, který ji bude stát několik tisíc měsíčně.

Firma bude nucena expandovat a částečná automatizace objednávek ji přijde vhod. Každý den zjistí přesný počet objednaných jídel na další den a veškeré adresy zákazníků. Předem si budou moci také naplánovat svoji rozvozovou trasu.

Tato malá ulehčení ušetří lidem ve firmě peníze i čas, který budou moci věnovat dalším aktivitám, což povede zvýšení produkce i zisku.

6.5 Celkové hodnocení

Implementovaný systém splňuje všechny počáteční požadavky firmy. Firma získá nástroj, díky kterému bude mít přehled o všech zákaznících. Do budoucna bych systém rozšířil o možnost procházení jednotlivých profilů uživatelů ze strany administrátora. Každý zákazník získá vlastní účet, přes který se bude do systému přihlašovat. Plusem systému je i to, zapomene-li uživatel své heslo, má možnost nechat si automaticky zaslat odkaz na svoji e-mailovou adresu, pomocí kterého si jej může změnit.

Pro objednání produktu uživateli stačí jen pár kroků – vybrat si produkt, časové období a objednat si. Poté už jen doma čeká na dovoz objednaných jídel. Veškeré vstupy v aplikaci jsou řádně ošetřeny a nemůže se stát, že si uživatel objedná produkt na nevhodné období. Podmínky ošetření se vyvíjí na základě dalších požadavků. Chce-li si uživatel odhlásit jídlo na daný den, i to je možné. Jednoduše může také měnit své osobní údaje a vyplnit svoji osobní poznámku, v které se mohou nacházet připomínky k objednávkám.

Z pohledu administrátora jistě mnoho funkcionalit chybí. V základu mu je umožněno vytvářet nové produkty, upravovat či je odstraňovat. Je především schopen zjistit veškeré údaje o objednávkách. Nedostatkem je chybějící možnost

výpisu objednávek pro zvolený den. V současném stavu je schopen získat přehled o objednávkách na den aktuální. Předpokládám, že tohle bude další z mnoha rozšíření systému.

Grafická podoba objednávkového systému by jistě chtěla doladit. Veškeré sekce jsou si docela podobné a na první pohled není patrné, kde se uživatel zrovna nachází. Na druhou stranu velké nadpisy a popisy mluví za vše a nemůže se stát, že by se uživatel ztratil. Především webové stránky, ale i systém by ocenily drobné animace pro lepší uživatelské vnímání (plynulé zvýraznění tlačítka po najetí myší, vysouvání responzivního menu nebo postupné zobrazování textu apod.). S touto uživatelskou přívětivostí souvisí i odesílání formulářů a dynamické změny vyžadující přístup k databázi. V současném stavu není použito technologie AJAX a určitě by bylo dobré tuto technologii v moderní aplikaci použít.

Velkou výhodou webové prezentace a objednávkového systému je responzivní řešení. Uživatelé, správci, kuchaři si je mohou otevřít na svém mobilním telefonu a nemusí mít přístup k počítači. Jsou podporovány všechny velikosti obrazovek. I to přispěje ke zvýšení konverze zákazníků.

Momentálně může uživatel nabývat v systému dvou rolí (běžný uživatel, administrátor). Dobré by bylo zamyslet se nad tím, zda nebude potřeba implementovat více rolí. Například kuchaři nemusí mít přístup ke správě produktů apod.

7 Závěr

Zadáním této práce bylo navrhnout a realizovat webovou prezentaci a objednávkový systém pro firmu Back2life. Tento systém má umožnit správcovství nabízených produktů a poskytovat možnost jejich snadného objednání.

První část práce je zaměřena na popis a vývoj webových aplikací, kde je popsáno, co je to webová aplikace, jaké se používají technologie a jak probíhá životní cyklus aplikace. Věnuje se také objektově orientovanému programování, a to především v jazyce PHP a vysvětluje, co je to PHP framework. Na tyto kapitoly navazuje část zabývající se frameworkem Laravel, kde jsou popsány jeho základní rysy.

Ve vlastní práci jsem se věnoval analýze požadavků firmy, z kterých vyšel patřičný výstup v podobě diagramů. Následovala samotná implementace webové prezentace a objednávkového systému a popis jeho základních funkcionalit.

Veškeré požadavky na projekt se podařilo splnit a jak webová prezentace, tak i objednávkový systém budou v nejkratší možné době spuštěny. Do té doby bude probíhat testování a hledání případných chyb a implementace dodatečných funkcionalit na základě žádostí zadavatele. V posledním kroku bude upřesněna cenová politika a bude implementována platební brána.

Na tomto projektu jsem si široce prohloubil své znalosti v programovacím jazyce PHP a seznámil jsem se, prostudoval a naučil se pracovat se zcela nový frameworkem Laravel. To považuji za svůj největší osobní přínos této práce.

Věřím, že i navrhnuté řešení objednávkového systému bude přínosné pro firmu a naplno jej využije. Spolupráce s firmou tímto okamžikem nekončí, ale bude pokračovat a bude se stále na objednávkovém systému pracovat. Budou se vyvíjet nové funkcionality a možnosti jak systém obohatit tak, aby firmě přinesl co nejlepší ohlasy a zasloužený užitek.

8 Literatura

1. *What are the Software Development Life Cycle (SDLC) phases?* 2015. ISTQB Certification Exam [online]. [cit. 2015-05-07]. Dostupné z: http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/
2. *Guide to Web Application Development*. 2015. Comentum [online]. [cit. 2015-05-07]. Dostupné z: http://www.comentum.com/guide-to-web-application-development.html
3. *HTML5*. 2014. TechTarget [online]. [cit. 2015-05-07]. Dostupné z: http://searchsoa.techtarget.com/definition/HTML5
4. *What Is JavaScript?* 2012. *Javascripter.net* [online]. [cit. 2015-05-07]. Dostupné z: http://www.javascripter.net/faq/whatisja.htm
5. *JQuery* [online]. 2015. [cit. 2015-05-07]. Dostupné z: https://jquery.com/
6. *What is AJAX?* 2015. Tutorialspoint [online]. [cit. 2015-05-07]. Dostupné z: http://www.tutorialspoint.com/ajax/what_is_ajax.htm
7. HAYDER, Hasin. *Object-Oriented Programming with PHP5: Learn to leverage PHP5's OOP features to write manageable applications with ease* [online]. 2007 [cit. 2014-12-19]. ISBN 978-1-847192-56-1.
8. *The MVC Pattern and PHP, Part 1*. 2015. Sitepoint [online]. [cit. 2015-05-07]. Dostupné z: http://www.sitepoint.com/the-mvc-pattern-and-php-1/
9. *ASP.NET MVC Tutorial*. 2015. W3schools.com [online]. [cit. 2015-05-07]. Dostupné z: http://www.w3schools.com/aspnet/mvc_intro.asp
10. *Web application framework*. 2014. DocForge [online]. [cit. 2015-05-07]. Dostupné z: http://docforge.com/wiki/Web_application_framework
11. *History of Laravel PHP framework, Eloquence emerging*. 2013. SURGUY, Maksim. Maxoffsky [online]. [cit. 2015-05-07]. Dostupné z: http://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/
12. *13 PHP Frameworks to Help Build Agile Applications*. 2014. Mashable [online]. [cit. 2015-05-07]. Dostupné z: http://mashable.com/2014/04/04/php-frameworks-build-applications/
13. ADERMANN, Nils a Jordi BOGGIANO. *Getting Started*. Composer [online]. [cit. 2015-05-08]. Dostupné z: https://getcomposer.org/doc/00-intro.md
14. LEWIS, Jason. 2013. *Your One-Stop Guide to Laravel Commands*. Tuts+ [online]. [cit. 2015-05-08]. Dostupné z: http://code.tutsplus.com/tutorials/your-one-stop-guide-to-laravel-commands--net-30349
15. SEVILLEJA, Chris. 2014. *A Guide to Using Eloquent ORM in Laravel*. Scotch [online]. [cit. 2015-05-08]. Dostupné z: https://scotch.io/tutorials/a-guide-to-using-eloquent-orm-in-laravel

16. MATHEWS, Sam. 2015. *Laravel's Query Builder and Eloquent ORM*. Cubet [online]. [cit. 2015-05-08]. Dostupné z: http://blog.cubettech.com/laravels-query-builder-and-eloquent-orm
17. *Routing*. 2014. CakePHP [online]. [cit. 2015-05-08]. Dostupné z: http://book.cakephp.org/2.0/en/development/routing.html
18. *Authentication with Laravel 5*. 2015. Tutsnare [online]. [cit. 2015-05-08]. Dostupné z: http://tutsnare.com/authentication-with-laravel-5/
19. OTWELL, Taylor. *HTTP Middleware*. Laravel [online]. [cit. 2015-05-08]. Dostupné z: http://laravel.com/docs/master/middleware
20. *Managing Databases with Migrations*. Laravelbook [online]. [cit. 2015-05-08]. Dostupné z: http://laravelbook.com/laravel-migrations-managing-databases/
21. *Database Seeding with Laravel*. Laravelbook [online]. [cit. 2015-05-08]. Dostupné z: http://laravelbook.com/laravel-database-seeding/
22. *Using Blade in Laravel 4*. 2013. BROWN, Philip. Culttt [online]. [cit. 2015-05-08]. Dostupné z: http://culttt.com/2013/09/02/using-blade-laravel-4/
23. ZORGDRAGER, Kelby. 2014. *What is OOCSS?* ApendTo [online]. [cit. 2015-05-09]. Dostupné z: http://appendto.com/2014/04/oocss/
24. HARISOV, Vitaly, Sergey BEREZHNOY a Vladimir GRINENKO. 2014. *Definitions*. BEM [online]. [cit. 2015-05-09]. Dostupné z: https://en.bem.info/method/definitions/
25. Keith Hindle, Debra Paul a Malcolm Eva. 2010. *Business Analysis*. Swindon, United Kingdom: British Informatics Society Limited. ISBN 978-1-906124-61-8.
26. BLAHA, M. UML *Database Modeling Workbook*. Westfield, U.S.A.: Technics Publications, LLC, 2013. 305 s. ISBN 978-1-9355045-1-1
27. GOSNEY, M. *PHP This! A Beginners Guide to Learning Object Oriented PHP*. [online]. 2013. URL: http://www.phpthis.com/
28. TAYLOR, O. *Laravel - The PHP framework for web artisans*. [online]. 2013. URL: http://laravel.com/
29. VO, J. *Laravel Book*. [online]. 2014. URL: http://learninglaravel.net/
30. SURGUY, Maksim. 2014. *Integrating date pickers*. Maxoffsky [online]. [cit. 2015-05-11]. Dostupné z: http://maxoffsky.com/code-blog/integrating-date-pickers/

9 Seznam obrázků

Přílohy