Česká zemědělská univerzita v Praze

Technická fakulta

Návrh a tvorba webové prezentace pro zadanou firmu

diplomová práce

Vedoucí diplomové práce:

Ing. Miroslav Mimra, MBA, Ph.D.

Autor:

Bc. Marek Pačes

PRAHA 2018

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Marek Pačes

Informační a řídicí technika v agropotravinářském komplexu

Název práce

Návrh a tvorba webové prezentace pro zadanou firmu

Název anglicky

Design and creation of websites for the specified company

Cíle práce

Cílem práce je navrhnout webovou prezentaci podle předem definovaných požadavků konkrétní společnosti.

Metodika

1. Vypracování rešerše
2. Zvolení cíle a metodiky diplomové práce
3. Vlastní práce – zjištění požadavků zadavatele, návrh a implementace webové prezentace
4. Diskuze a závěr

Doporučený rozsah práce

50 – 60 stran

Klíčová slova

webová prezentace, optimalizace, tvorba webových stránek

Doporučené zdroje informací

CLIFTON, B. *Google Analytics : podrobný průvodce webovými statistikami.* Brno: Computer Press, 2009. ISBN 978-80-251-2231-0.

DARIE, C. *AJAX a PHP : tvoříme interaktivní webové aplikace profesionálně.* Brno: Zoner Press, 2006. ISBN 80-86815-47-1.

HOGAN, B P. *HTML5 a CSS3 : výukový kurz webového vývojáře.* Brno: Computer Press, 2011. ISBN 978-80-251-3576-1.

NEUMAJER, O. *Budujeme školní web.* Brno: CP Books, 2005. ISBN 80-251-0612-8.

PROKOP, M. *CSS pro webdesignery.* Praha: Mobil Media, 2003. ISBN 80-86593-35-5.

RAHMEL, D. *Joomla : podrobný průvodce tvorbou a správou webů.* Brno: Computer Press, 2010. ISBN 978-80-251-2714-8.

SALIM, F. – ALBERS, B. – LUBBERS, P. *HTML5 : programujeme moderní webové aplikace.* Brno: Computer Press, 2011. ISBN 978-80-251-3539-6.

TEAGUE, J C. *DHTML a CSS pro World Wide Web : praktická vizuální příručka.* Praha: Softpress, 2005. ISBN 80-86497-77-1.

Předběžný termín obhajoby

2017/18 LS – TF

Vedoucí práce

Ing. Miroslav Mimra, Ph.D.

Garantující pracoviště

Katedra využití strojů

Elektronicky schváleno dne 17. 1. 2017

doc. Ing. Petr Šařec, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 1. 2017

prof. Ing. Vladimír Jurča, CSc.

Děkan

V Praze dne 31. 03. 2018

Čestné prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Miroslava Mimry, MBA, Ph.D. a uvedl v přiloženém seznamu veškerou použitou literaturu i další zdroje.

Jsem si vědom, že odevzdáním diplomové práce souhlasím s jejím zveřejněním dle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů, ve znění pozdějších předpisů, a to i bez ohledu na výsledek její obhajoby.

Jsem si vědom, že moje diplomová práce bude uložena v elektronické podobě v univerzitní databázi a bude veřejně přístupná k nahlédnutí. Jsem si vědom že, na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, ve znění pozdějších předpisů, především ustanovení § 35 odst. 3 tohoto zákona, tj. o užití tohoto díla.

V Praze dne ……..

……..

Bc. Marek Pačes

Poděkování

Tímto děkuji vedoucímu mé diplomové práce Ing. Miroslavu Mimrovi, MBA, Ph.D. za poskytnutí možnosti vypracovávat u něj zvolené téma a za cenné rady k formulaci práce.

Abstrakt: Cílem práce je navrhnout a vytvořit webovou prezentaci pro podle předem definovaných požadavků. Vytvořit vhodné prostředí pro budoucí systémový rozvoj a navrhnout další kroky, které by se měly realizovat. Před samotnou realizací proběhne popis nástrojů a technologií, které lze použít v praktické části. Následně budou vybrány ty, které nejvíce vyhovují zadaným náležitostem a proběhne vlastní realizace společně se základní optimalizací. Tím vznikne nový nástroj s potenciálem zvýšit odbyt společnosti, pro kterou je prezentace tvořena, zároveň centralizuje dokumentaci a zvýší přehled o aktuálním stavu.

Klíčová slova: webová prezentace, optimalizace, tvorba webových stránek.

Design and creation of websites for the specified company

Summary: The aim of the thesis is to design and create a web presentation with pre-defined requirements. Create an appropriate environment for future system development and suggest further steps to be done. Before implementation, a description of the tools and technologies that can be used in the practical part will be provided. Subsequently, those that most suits to given requirements will be selected and their own implementation together with the basic optimization will take place. This create a new tool with the potential to increase the sales of the company for which the presentation is created, with the same time centralize documentation and increase the current status overview.

Keywords: web presentation, optimization, website creation.

Obsah

1 Úvod

Webová prezentace představuje jeden ze základních nabídkových nástrojů, který lze ke zvýšení konkurenceschopnosti na trhu využít. Opomíjení tohoto nástroje je nevyužitím prodejního kanálu s širokou základnou potencionálních zákazníků. Abychom tento kanál využili co možná nejvíce, je vhodné realizovat uživatelsky přívětivé a funkční prostředí, ve kterém můžeme naši nabídku prezentovat. Samozřejmostí je optimalizace tohoto prostředí tak, aby bylo správně prezentováno na všech zařízeních se schopností toto prostředí využívat a zároveň bylo snadno dostupné.

Z tohoto důvodu bude v této práci realizována webová prezentace, která má tento potenciál využít a vytvořit tak nejen prostředí pro nabídku, ale také pro provozovatele samotného. Ten díky němu mimo jiné získá nový nástroj, kterým může evidovat aktuální stav jeho nabídky.

Provozovatelem bude chovatel mastného skotu plemene charolais a burských koz, který aktuálně nevyužívá možnosti nabízet své produkty prostřednictvím webové prezentace. Zároveň chovatel vlastní řeznictví, ve kterém nabízí své produkty, takže i toto by mělo být v prezentaci zahrnuto.

2 Cíl práce

Cílem práce je navrhnout webovou prezentaci podle předem definovaných požadavků konkrétní společnosti. Dílčím cílem je pak příprava systému pro další rozšíření a základní optimalizace.

3 Metodika práce

První část práce bude zaměřena na přehled a popis technologií, pomocí kterých lze webovou prezentaci vytvořit. Bude obsahovat ty aktuálně nejrozšířenější technologie společně s požadavky, které kladou na programové vybavení. Kapitoly zabývající se preprocesory budou doplněny o krátké praktické ukázky, které napomohou čtenáři lépe pochopit význam jejich aplikace. Jako podklady budou sloužit ověřené zdroje a odborná literatura. Dále autorovy vlastní znalosti a zkušenosti. Ty budou aplikovány zejména v kapitolách, ve kterých budou popsány technologie, s nimiž je nejvíce ve styku.

V praktické části bude přestaveno souhrnné zadání s požadavky na vlastní realizaci. Proběhne zvolení výběrových kritérií a určení jejich vah. Podle těchto kritérií se vyberou vhodné technologie. Bude představen grafický návrh a následně definován způsob, jakým bude řešena problematika prezentace na různých zařízeních. Vytvoří a nastaví se pracovní prostředí s jeho závislostmi. V samotném systémovém řešení bude uvedena adresářová struktura systému a princip jakým systém funguje a prezentuje data. Poté budou v jednotlivých podkapitolách rozebrány jednotlivé modulární části, jejich databázové návrhy a důležité funkce. Závěr kapitoly předvede výsledky stávající optimalizace na rychlosti načítání v prohlížeči.

V úplném závěru budou shrnuty a zhodnoceny realizované činnosti včetně jejich přínosu. Budou také uvedena závěrečná doporučení, která mají za cíl určit směr budoucího vývoje.

4 Přehled dostupných technologií

Nejprve je třeba seznámit se s aktuálními technologiemi a nástroji, pomocí kterých lze webové stránky tvořit. Kapitola je zaměřena na jejich popis, případně výhody a nevýhody společně se základy použití.

Při realizování webových stránek je vhodné rozlišovat dva pohledy, pomocí kterých lze lépe chápat technologické rozdělení jednotlivých oblastí v dané problematice. Prvním je pohled programátora stránek, druhým pak pohled kodéra. Jedná se o dva rozdílné obory, které spolu sdílí využívané technologie, případně si poskytují jejich výstupy. Mezi technologie se řadí i frameworky a používané jazyky.

Kapitoly spadající do kodérských činností se zabývají jazykem HTML5 a jeho podporou v prohlížečích, jazykem CSS3 s jeho hlavními vlastnostmi, CSS frameworky a preprocesory. Mezi ty programátorské spadají kapitoly o PHP, ASP.NET, JavaScriptu a PHP frameworcích. Ve společném poli působí redakční a šablonovací systémy.

V jednotlivých kapitolách jsou tak jednoduše popsány vybrané technologie, které mohou být použity pro vlastní zpracování zadané práce.

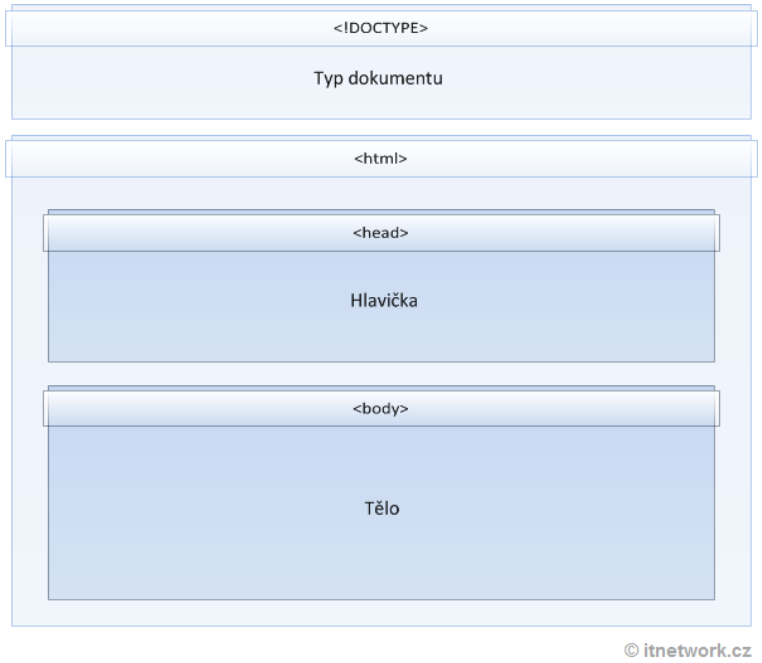
4.1 HTML5

Hypertextový značkovací jazyk, známější pod zkratkou HTML (HyperText Markup Language) tvoří společně s kaskádovými styly neboli CSS (Cascading Style Sheets), JavaScriptem a serverovým jazykem PHP (Hypertext Preprocessor) sadu základních nástrojů pro tvorbu webových stránek. [1]

V této kapitole bude popsána jeho pátá verze, která je současně verzí aktuální a přináší některé změny v sémantických významech u starších HTML tagů, ale hlavně zavádí značky nové. [2]

Samotný jazyk HTML vznikl na počátku 90. let minulého století ve velmi omezené podobě, kdy popisoval několik málo elementů používaných pro tvorbu webových stránek. [3]

Obecně lze říci, že jazyk HTML slouží k popisu toho, co webová stránka obsahuje, při dodržování základních standardů definovanými konsorciem W3C (World Wide Web Consortium). Definuje také základní strukturu stránky pomocí neutrálních značek * * a **, které spojují elementy na stránce do logických celků pro následné aplikování vzhledu jazykem CSS. Tato struktura vždy vychází z následujícího tvaru viz Obr. 4.1. [4] [5]



Obr. 4.1 Struktura HTML dokumentu [6]

V HTML kódu může být podoba následovná.

```
<!DOCTYPE html>
<html>
        <head>
                <title>Diplomová práce</title>
        </head>
        <body>
                <h1>Název diplomové práce</h1>
                <p>Popis diplomové práce</p>
        </body>
</html>
```

HTML5 přináší mnoho nového, ale stále zachovává zpětnou kompatibilitu. Novými, validními, sémantickými prvky jsou například: *<article>* (článek), *<section>* (část), *<header>* (záhlaví), *<footer>* (patička*), <figure>* (obrázek).

Přičemž *<section>*, *<header>* ani *<footer>*, nepředstavují obalový element. Neměly by se tedy používat jako kostra pro stylování jednotlivých bloků webových stránek. Místo toho je jejich použití vhodné tam, kde to dává sémantický význam.

Definování typu formulářových polí je další novinkou. Typů je dohromady více jak tucet. Jednotlivé typy by měli odpovídat tomu, jaká data pole očekává od uživatele. Pokud má pole nastaven typ na číslo (number) a jsme na mobilním zařízení, vyskočí nám rovnou klávesnice s čísly. Stejně tak u typu datum (date), kde pole vytvoří masku pro zadání ve správném formátu. Moderní prohlížeče společně s tím nabídnou klikací kalendář. [7]

HTML5 přináší nativní podporu videa, kdy není potřeba externích pluginů. Stačí tak do stránek vložit značku *<video>* s adresou souboru. Lze také nastavit automatické přehrávání, smyčku, ztlumení nebo zobrazení ovládacích prvků. [8]

Element *<canavas>* nabízí možnost kreslení grafiky pomocí JavaScriptu. Můžeme pracovat s 2D nebo 3D kontextem. Vytvoříme plochu, která reprezentuje bitmapové kreslící plátno. To může být používáno pro vykreslování grafů, herní grafiky a dalších vizuálních prvků. [9] [10]

Modernizr řeší situaci, kdy prohlížeč nepodporuje danou technologii. Je to JavaScriptová knihovna, která umí detekovat řadu vlastností HTML5 a CSS3. Knihovna se umístí do hlavičky HTML dokumentu a již není potřeba jí dále volat (inicializovat). Na stránce vytvoří globální objekt s názvem Modernizr, ten obsahuje sadu logických (boolean) funkcí. Pokud tak prohlížeč neumí například zmíněný *<canavas>*, bude hodnota vlastnosti Modernizr.canvas rovna jedné (true). [7]

```
if (Modernizr.canvas) {
  // kreslení podporováno
} else {
  // bez podpory
}
```

4.2 CSS3

První verze tohoto jazyka se objevily až několik let po vzniku samotného HTML, oficiálně se jazyk prosadil v roce 1996. Stejně jako u HTML5 se jedná o přirozené rozšíření předchozích verzí. U jazyka CSS byl pokrok mezi jeho verzemi dosti značný a aktuální třetí verze je již plnohodnotný nástroj pro tvorbu vizuální podoby stránky. [3]

Šablonou stylů je obyčejný textový soubor, jenž obsahuje soubor pravidel, který má definované vlastnosti a hodnoty určující chování a vzhled elementů v HTML. Stavba těchto pravidel je složena ze selektoru, ten určuje, na jaké elementy se dané pravidlo vztahuje a dále z deklaračního bloku. Deklarační blok se skládá z jedné či více dvojic vlastnost-hodnota, přičemž tyto dvojice tvoří tzv. deklaraci, která specifikuje samotné chování elementu. [3]

Hodnoty vlastností mohou být číselné, jiné zase přijímají určitou vlastnost z předem definovaného seznamu, URL adresu nebo například barvu. Jednou z nových možností v CSS3 je možnost používat definici barev ve formátu RGBA, u kterých je oproti RGB přidán alfa kanál, pomocí kterého se nastaví míra průhlednosti. Další novinkou jsou vlastnosti pro definici zaoblených rohů, přechodů nebo více obrázků na pozadí. Nově lze také zapisovat matematické výrazy. [3] [11]

Tím se dostáváme k problematice kompatibility u jednotlivých verzí prohlížečů. Například u uvedeného zápisu barvy ve formátu RGBA dochází ke špatnému zobrazení v prohlížeči Internet Explorer (před verzí 9). Nejlépe podporovanou verzí jazyka CSS tak stále zůstává CSS2. Vhodným nástrojem ke zjištění kompatibility je nástroj „Can I use“, dostupný na stránkách www.caniuse.com. [7]

4.2.1 Flexbox

V oblasti pozicovaní elementů na stránce patří flexbox k revoluční metodě. Jeho základním pravidlem je, že není žádný přesně definovaný směr nebo pořadí, ve kterém se mají elementy vykreslovat. Sám kodér si tak může určit směr řazení prvků (horizontálně, vertikálně, zleva, zprava, shora, zdola). Jelikož není pevně dáno ani pořadí, může prvky přesouvat, a to nezávisle na definici HTML. [12]

Prvky se ve flexboxu chovají podobně, jako buňky tabulky a plovoucí prvky. Pokud se nevejdou do určeného prostoru, mohou se zalomit, a tím tak vytvořit nový řádek či sloupec. Umí také vyplňovat zbývající prostory, jako to dělá například blokové zarovnání textu. Využití flexboxu je tak ideální v případě, že chceme adaptivní rozložení stránky, které je schopno automaticky reagovat na případné velikostní změny, ať už sama sebe, nebo velikosti okna. [12]

4.2.2 Media Queries

Řeší adaptovatelnost stránek na různých zařízeních, resp. na různých velikostech počítačových monitorů a mobilních displejů. Jedná se o tzv. responzivní design. Dotazy na médium jsou definované v modulu Media Queries, které rozšiřují typy médií o syntaxi, pomocí které lze přesněji specifikovat jaké zařízení uživatel používá. [13]

Lze tak stránky přizpůsobit různým rozlišením. V případě mobilních telefonů a tabletů pak také orientaci obrazovky, případně poměr CSS pixelů k těm hardwarovým. To se hodí pro detekci retina displejů.

```
@media (-webkit-min-device-pixel-ratio: 2), (min-resolution: 192dpi)
{
    /* retina specifikace */
}
```

4.2.3 Prefixy

Správné zobrazení stylů u jednotlivých prohlížečů, řeší CSS prefixy. Nové vlastnosti, které se přidávají při vývoji prohlížečů, mohou mít odlišnou interpretaci ve finální verzi. Výrobce si tak k vlastnosti přidá svůj unikátní prefix viz Tab. 4.1. Odpadá tak riziko, že finální podoba vlastnosti poškodí již dříve funkční stránky. Zároveň je zřetelné, že prohlížeč vlastnost podporuje. [13]

Tab. 4.1 Seznam používaných prefixů [14]

prefix	prohlížeč
-webkit-	Google Chrome, Safari, Android Browser
-moz-	Firefox
-o-	Opera
-ms-	Internet Explorer, Edge

Možností aplikace je hned několik, tím méně šťastným je způsob ručního vypsání. V takovém případě musíme vědět, kdy prefixy potřebujeme, nebo se dotázat na www.shouldiprefix.com. Druhou možností je použít některý z autoprefixerů, který zajistí prefixování za nás. V případě, že je použit některý z preprocesorů, může se na prefixy použít mixin. Ty jsou popsané v následující kapitole.

4.3 CSS preprocesory

Jedná se o dynamické jazyky pro tvorbu stylesheetů, které rozšiřují CSS o prvky, jako jsou proměnné, mixiny, výpočty a funkce. Jednotlivé preprocesory se liší zejména svou syntaxí. Výstupem je u všech CSS, kterému prohlížeč rozumí. Oproti klasickému CSS nám nově přináší například proměnné, mixiny, vnořená pravidla nebo výpočty a funkce. Jejich výhodou tedy není jen zkrácení zdrojových kódů, ale také přiblížení se programovacím jazykům. Nyní budou uvedeny tři nejčastěji používané preprocesory, kterými jsou LESS, SASS a Stylus. [15]

4.3.1 LESS

Ze všech preprocesorů LESS nejvíce respektuje deklarativní povahu jazyka CSS, snaží se tedy co nejvíce respektovat jeho původní filozofii. To je však na úkor elegantnosti zápisu u pokročilých technik, jako jsou cykly nebo podmínky. [16]

Výstupem je klasické CSS, kterému prohlížeč na rozdíl od LESS souboru rozumí. Tento princip je stejný u všech preprocesorových jazyků. Aby bylo možné převést LESS soubor do CSS, je

nutná jeho kompilace. Tuto kompilaci je možné spustit jak na klientské straně, tak na straně serveru.

V případě kompilace v prohlížeči klienta je nutné připojit do hlavičky HTML dokumentu knihovnu LESS.js, která kompilaci zajistí. Tato metoda nemusí být úplně vhodnou, pokud například prohlížeč nebude podporovat JavaScript. Zároveň se tím prodlužuje doba načtení samotné stránky. [15]

Druhou možností je tedy kompilace na serveru, kde je možnost využít instalaci serverového frameworku Node.js a pohodlně tak vše na lokálním serveru zkompilovat před samotným nasazením na produkční webové stránky. Další variantou kompilace jsou specializované programy s grafickým uživatelským rozhraním (GUI), jako jsou Prepros, Crunch 2 nebo Koala. Ty spolu s funkcí kompilace nabízejí možnosti minifikace výsledného CSS souboru nebo automatické doplnění prefixů.

Proměnné jsou jedním ze základních rozšíření, které preprocesor nabízí. Následuje ukázka, jak vypadá kód před a po kompilaci.

```
@barva-hlavni: #fff;
#zahlavi {
  color: @barva-hlavni;
}
```

Po kompilaci má výsledný kód podobu klasického CSS.

```
#header { color: #fff; }
```

V LESS lze vkládat do sady pravidel různé sady vlastností z jiné třídy, které jsou definované v kódu. Sada se tak může aplikovat v jakékoli další třídě. V příkladu se sada aplikuje na obě třídy. [15]

```
// sada pravidel třídy vlastností
.podtrzeni {
      text-decoration: underline;
      color: red;
}
```

```
// vložení sady do jiné třídy
a.odkaz {
  .podtrzeni;
}
```

Parametrické mixiny jsou speciálním typem pravidel, které akceptují parametry a je zde možnost jejich využití jako mixinů tříd. [15]

```
// mixin s předdefinovaným parametrem
.pozadi (@barva: #fff) {
  background: @barva;
}
// předání parametru (změna barvy)
body {
  .pozadi(#000);
}
```

Pro zvýšení přehlednosti kódu a zjednodušení zápisu dlouhých selektorů, slouží vnořování pravidel.

```
.rodic {
  background: @barva;
  .potomek {
    &:hover {
      background: darken(@brava, 10%);
    }
  }
}
```

V přikladu si lze všimnout také funkce pro ovládání barev, zde je to ztmavění barvy o 10 %. V nabídce je mnoho dalších funkcí, které řeší například saturaci a desaturaci barev, velikost alfa kanálu (průhlednost), kontrast a další. [17]

Významnou funkcí nejen jazyka LESS, ale i ostatní preprocesorů, je importování zdrojových souborů do sebe. Tím vzniká možnost vytvářet například parciální komponenty v samostatných souborech, které se pomocí importu vloží do výsledného kódu. Rozlišuje se import klasický a referenční. V klasickém importu se vezme veškerý obsah a vloží na požadované místo. U referenčního se nikam nic nevkládá a nevzniká tak ani požadavek, který by kompilaci zpomaloval. Pomocí referenčního importu lze rozšiřovat třídy o vlastnosti, které se nachází v jiném souboru. [18]

```
@import (reference) "tlactika-barevna.less";
.tlacitko {
  &:extend(.tlacitko-modre all);
}
```

4.3.2 Sass

Další z nabídky preprocesorů využívá syntaxi Sass, známou jako odsazovaná syntaxe a SCSS (Sassy CSS). Syntaxe Sass se spoléhá na odsazování, odstranění závorek, středníků a ostatních symbolů, takže výsledný kód je stručnější. Naopak SCSS má blíže ke klasickému CSS. [19]

```
// Sass
$barva-hlavni: #fff;
#zahlavi
  color: $barva-hlavni

// SCSS
$barva-hlavni: #fff;
#zahlavi {
  color: $barva-hlavni;
}
```

Proměnné jsou uvozené klíčovým znakem $, následuje název proměnné, dvojtečka a hodnota. Oproti syntaxi LESS tak používá jiný klíčový znak, výstupem je opět klasické CSS. Sass však pohlíží na proměnné odlišně v oblasti deklarování lokálních a globálních proměnných, kdy lokální hodnota přepisuje globální a všechny proměnné definované mimo selektory jsou považovány za globální. Sass je název preprocesoru, proto je tak nazýván i v následujících komentářích v kódu, ačkoli syntaxe je SCSS.

```
$barva: black;
.prvni {
  $barva: white;
  color: $barva;
}
.druhy {
  // LESS = black (vypíše globální proměnnou)
  // Sass = white (přepsáno lokální proměnnou z třídy .první)
  color: $color;
}
```

Problematiku lokálních a globálních proměnné řeší používání tzv. flags. Umístění *default flag* (!default) na konec proměnné bude mít za následek, že pokud již byla proměnné přiřazena hodnota, nebude již následně přepsána. Druhou možností je *global flag* (!global). Jak už název napovídá, jejím použitím se změní rozsah proměnné na globální a je tak možno s ní tak pracovat napříč kódem. [20] [21]

Rozšiřování selektorů řeší funkce *@extend*. Výhodou rozšiřování je, že se nevkládá (neduplikujeme) kód z jiné třídy, ale přidá se selektor třídy, ve které *@extend* použijeme. [22]

```
// Sass
.prvni {
  /* Sass kód */
}
.druha {
  /* Sass kód (unikátní) */
  @extend .prvni;
}
// CSS
.prvni,
.druha {
  /* Sass kód */
}
.druha {
  /* Sass kód (unikátní) */
}
```

Mezi nejvíce používané funkce celého Sassy patří mixiny. Na příkladu je uveden jeden, který bude umět přijímat argumenty a jeho výstupem bude definice šířky a výšky elementu zároveň. [19]

```
@mixin velikost($sirka, $vyska: $sirka) {
  width: $sirka;
  height: $vyska;
}
```

4.3.3 Stylus

Stylus, podobně jako odsazovaná sytaxe Sass dělá kód stručnějším. Dovoluje vynechávání závorek a středníků. Dokonce úplně odstraňuje označení proměnných. Kód pak může vypadat například takto.

```
.prvni
.druhy
  color black
```

Snaží se tedy o to, aby kodér psal co možná nejméně kódu. K tomu mu má pomoci i knihovna NIB, obsahující mixiny, utility, komponenety a řešení barevných přechodů. Takto, pro porovnání se Sass, vypadá utilita z této knihovny, která řeší velikost elementu. [23]

```
velikost()
  if length(argumenty) == 1
    width: argumenty[0]
    height: argumenty[0]
  else
    width: argumenty[0]
    height: argumenty[1]
velikost()
if length(argumenty) == 1
     width: argumenty[0]
     height: argumenty[0]
else
     width: argumenty[0]
     height: argumenty[1]
```

Zde je již využita rozhodovací logika, která podle délky pole rozhodne, zda kodér zapsal hodnotu pro šířku i výšku, nebo použije jednu z nich. V takovém případě vytvoří čtverec.

4.4 CSS frameworky

Existuje řada frontend frameworků s různou komplexností. Některé řeší pouze mřížku (grid) stránky, jiné v sobě zahrnují celé komponenty včetně JavaScriptů. Lze tak značně urychlit vývoj stránek nebo framework použít pouze na prototypování.

4.4.1 Bootstrap

V současné době se vyskytuje mnoho frontend frameworků, nicméně Bootstrap se mezi nimi prosadil nejvíce. Obsahuje sady HTML, CSS a JavaScript komponent. Ty pomáhají u základního pozicovaní elementů na stránce pomocí upravitelného gridu, nabízí pomoc při realizaci responzivního řešení, typografie nebo s formuláři. Nabízí se také jako nástroj pro prototypování stránek. [24]

Jeho aplikace je vhodná i na administrační část, kde lze tyto komponenty aplikovat. Představuje ideální kombinaci s CSS preprocesory, kdy se kompilují jen ty části (komponenty) frameworku, které jsou pro práci potřeba. [25]

Výhodou Bootstrapu je jeho velká komunita. Na serveru github.com má více jak 120 000 hvězdiček, oproti tomu Foundation má pouze přes 27 000. [26] [27]

4.4.2 Foundation

Druhým známým zástupcem je Foundation. Oba dva využívají dvanáctisloupcový grid a oba dva razí „mobile first“ přístup (tedy kódování od mobilního rozlišení výše) a i licenční politiku mají stejnou (MIT licence). Podobné je to u nabízených předpřipravených komponent (navigace, tlačítka, stránkování a další). [28]

Foundation nabízí zdrojové kódy v preprocesoru Sass, naopak Bootstrap byl ve své 3. verzi vyvíjen v Less, ale je možné stáhnout i oficiální port na Sass. Ve 4. verzi je již také vyvíjen v Sass. Foundation používá CSS jednotky *rem*, Bootstrap do 3. verze používal *pixely*, ve 4. verzi pak také *rem*. [29]

Mezi klady patří lepší prostředí pro vlastní úpravy, není potřeba přidávat elementům speciální třídy zajišťující responzivitu a také fakt, že vzhledem k popularitě Bootstrapu vzniká jistá unifikace, kterou může Foundation rozbít. [28]

4.5 PHP

Skriptovací jazyk pracující na straně serveru, který je určen pro tvorbu dynamických internetových aplikací. Je objektově orientovaný (nemusí tak být používán) a jeho syntaxe je podobná jazyku C. Patří mezi ty jazyky, u kterých se nemusí předem definovat typ proměnných. Snadno komunikuje s databázemi, jako je například MySQL. Je multiplatformní, takže jej lze provozovat na většině webových serverů a operačních systémů. Nabídka poskytovatelů webhostingových služeb je tak veliká. [30]

Naopak nevýhodou může být, že je to jazyk interpretovaný, ne kompilovaný. Pokud má někdo přístup do serveru, pak má přístup i do kódu samotného. [31]

4.6 ASP.NET

Je sada knihoven postavena na .NET frameworku, s .NET sdílí společné CLR (společný jazykový modul runtime), a tak tato sada umožňuje programování v ruzných jazycích (např. Visual Basic .NET, JScript.NET, C#). Tato technologie je založena na principu klient-server. ASP běží na serverové straně a výstupem je HTML. Díky kompilaci kódu běží aplikace rychleji a chyby mohou být zachyceny již při vývoji. [32]

Pro běh ASP.NET je primárně určena platforma Windows, existují ale i podpora pro Unix, například projekt Mono sponzorovaný právě Microsoftem. [33]

Nová verze ASP.NET Core již běží i na platformách Mac OS X a některých distribucích Linuxu. Cílem je vytvořit prostředí nezávislé na specifické platformě. Díky knihovnám .NET Standard je zachována kompatibilita s .NET Frameworkem nebo se zmíněným Mono. [34]

4.7 JavaScript

Server s běžícím Node.js a aplikace psaná v JavaScriptu (běžícím na straně serveru), tak může vypadat moderní způsob tvorby webových stránek a aplikací. Node.js je dosti výkonné, událostmi řízené prostředí pro JavaScript. Základ tvoří interpreter V8 z Google Chrome společně s trochou kódu v C++. Pro běh prostředí Node.js je využíváno balíčkovacího systému NPM. Obě tyto technologie je však možné využívat i pro automatizaci kodérských činnosti. Pomocí NPM instalujeme Node programy a knihovny, včetně závislostí.

Pro tvorbu single-page aplikací (existuje jedna stránka, která zbytek načítá přes AJAX) může být dobrou volbou použití frameworku AngularJS. Problémem u stránek napsaných v JavaScriptu by mohla být optimalizace pro vyhledávače (SEO). Správné indexace dosáhneme například pomocí HTML5 History API, takže tento problém je řešitelný. Pro Node.js je jednou z nejoblíbenějších databází MongoDB.

4.8 Redakční systémy

Redakční systémy, často označovány zkratkou CMS (content management system), tedy systémy pro zprávu obsahu, jsou při realizací webových stránek mnohdy vhodnou alternativou, před realizací nebo použitím systému vlastního. Existuje jich celá řada, mezi neznámější trojici řadíme WordPress, Joomla! a Drupal. Všechny jsou řešeny jazykem PHP a (typicky) využívají databázi MySQL. Výhodou často je, že zákazník již může znát administrační rozhraní takových systémů a je pro něj tedy snazší správa jeho stránek.

Existuje celá řada rozšíření (pluginů), které zmíněné CMS nabízí a jsou zcela zdarma. Při vývoji webových stránek, tak lze základní funkce, které mají stránky plnit, sestavit bez nutnosti financování vlastního vývoje. Ten přijde na řadu tehdy, když chceme stránky customizovat (upravit dle požadavků zákazníka) nad rámec nabízených rozšíření.

4.8.1 WordPress

Tento open sourcový projekt (licence GNU GPL) nabízí vývojáři možnost zapojení se do vývoje a vytvářet tak rozšíření *na míru*. Dostupný je již od roku 2003 a od té doby se stal velice populárním i vzhledem k počtu rozšíření a faktu, že drtivá většina je dostupná zdarma. Jeho otevřená licence dovoluje vývojáři nejen upravovat nebo dále šířit systém, ale i fakturaci za odvedenou práci. Základní aktualizace i aktualizace bezplatných rozšířeních jsou vždy zdarma. Rozšíření jsou tvořena tak, aby jejich administrační prostředí byla sjednocená v duchu základní administrace a výsledek působil kompaktním dojmem. [35] [36]

Na druhou stranu, některá rozšíření nejsou kvalitní. Je mnoho programátorů *na volné noze*, kteří si tvorbou rozšíření přivydělávají a výsledek pak nemusí být v požadované kvalitě, kterou by jinak nabídla komplexní realizace od jedné firmy. Často se také zákazníci snaží si jejich stránky do co možná největší míry udělat sami, výsledkem je pak systém sestavený z mnoha rozšíření, bez konceptu a s jinou funkčností, než požadují. Rizikem jsou také samotné aktualizace, které můžou způsobit komplikace například ve chvíli, kdy přes redakční systém (jeho rozšíření) provozujeme internetový obchod. Z pohledu bezpečnosti je na tom samotné jádro systému dobře, rizikem ale mohou být právě jeho rozšíření. Podporované databáze jsou MySQL a MariaDB. [36] [37]

4.8.2 Joomla!

Oproti CMS WordPress je mladší, první verze debutovala v roce 2005, ale její vývoj sahá až do roku 2000. V České republice se tomuto CMS dostává velké popularitě, zejména díky početné a aktivní skupině komunitních vývojářů. Začátkem roku 2018 se v databázi rozšíření nachází necelých 8 000 rozšíření. Jeho licence je stejná jako u CMS WordPress, je tedy zdarma a může se volně šířit. Funkční základ je podobný tomu, co nabízí WordPress. Tedy vytváření článků, jejich řazení, vytváření uživatelských účtů, které můžeme řadit do skupin s definovanými právy atp. Joomla! není vázána pouze na databázi typu MySQL, podporované jsou také PostgreSQL a Microsoft SQL Server. [38] [39] [40]

4.8.3 Drupal

Je dalším zástupcem CMS, rovněž s licencí GPU GPL, který byl vydán jako open source v roce 2001. Jeho rozšíření mezi populární CMS započalo v roce 2011, kdy byla vydána sedmá verze. S ní přišlo mnoho vylepšení, lepší podpora obecných webových standardů, uživatelská přívětivost, vícejazyčnost a bezpečnost. Právě bezpečnost je jeho velkou devizou. Dobrou pověst získal díky organizovanému procesu vyšetřování, ověřování a následného publikování potencionálních bezpečnostních problémů. Začaly ho tak používat i světové vládní instituce. V České republice bylo prvním velkým úspěchem nasazení na stránkách televize FTV Prima, spol. s r. o. a následně u veřejnoprávních institucí České televize a Českého rozhlasu. [41]

Drupal se postupně proměňuje z klasického redakčního systému spíše na framework, na kterém lze postavit v podstatě jakoukoli realizaci na míru požadavkům. Komplexně řeší propojení datových toků různých systémů a více odděluje logickou vrstvu správy stránek od prezenční vrstvy. Tím vzniká možnost správy prostřednictvím jiných systémů, jako jsou například hybridní televize nebo mobilní aplikace. [42]

4.9 PHP frameworky a šablonovací systémy

Před realizací vlastního systému je dobré se rozhodnout je potřeba vybrat vhodné nástroje, aby byla vlastní práce kvalitní a efektivní. Existuje řada frameworků a šablonovacích systémů (jazyků), které nám s prací pomohou. Následuje výčet s významnými zástupci.

V roce 2004 došlo (po příchodu nového jádra Zend Engine 2) k přepracování objektového modelu v PHP 5.0, ten dal vzniknout velkému množství open source objektově orientovaných frameworků s MVC (model-view-controller) architekturou. [43]

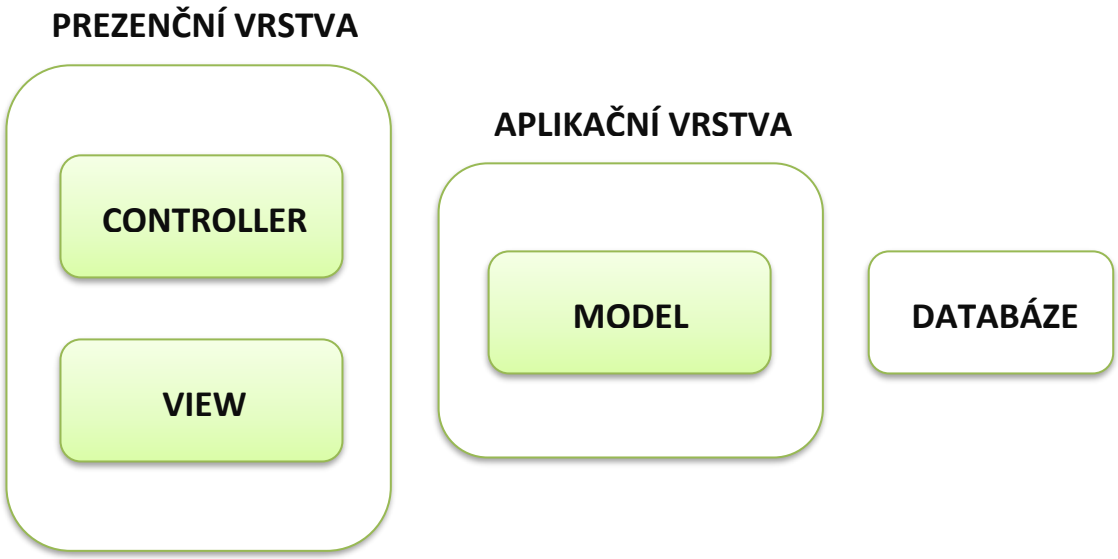
MVC architektura vznikla, aby byl kód obsluhy (*controller*) oddělen od kódu aplikační logiky (*model*) a také od kódu zobrazujícího data (*view*). Tím dojde k zpřehlednění a usnadnění budoucího vývoje společně s možností testovat jednotlivé části zvlášť.

Model představuje funkční základ stránek, obsahuje aplikační logiku. Všechny uživatelské akce, představují akce *modelu*. Ten pak ven nabízí pevně dané rozhraní. Voláním tohoto rozhraní, resp. jeho funkcí lze zjišťovat nebo měnit jeho stav, který si sám spravuje. *Model* je od *view* a *controlleru* oddělen a neví tak o jeho existenci.

View neboli pohled je vrstva, která má na starost zobrazování výsledků jednotlivých požadavků. Právě v této vrstvě se typicky používá šablonovací systém, který umí zobrazit výsledky z modelu.

Controller je řadič, zpracovávající požadavky uživatele, podle kterých volá příslušnou aplikační logiku, tedy *model*. Následně požádá *view* o vykreslení dat. [44]

S použitím šablonovacího systému vznikne optimální prostředí pro vývoj. Čisté PHP v prezenční vrstvě, která je součástí modelu MVC (viz Obr. 4.2) přináší jistá rizika. Těm je možné se vyvarovat právě použitím šablonovacího systému.



Obr. 4.2 Seskupení vrstev v MVC modelu [56]

Zjednodušení syntaxe, časté opakování částí kódu, bezpečnost, vše řeší šablonovací jazyky. U bezpečnosti je to zejména *escapování* obsahu proměnných z důvodu ochrany proti útokům XSS (cross site scripting). Šablonovací systémy se vyskytují ve formě znovupoužitelných knihoven (např. Smarty) nebo jako integrované součásti frameworků (např. Nette). [43]

Další vlastností je dědičnost, jejíž přímá podpora v PHP chybí, řeší jí tak šablonovací systémy. Pomocí dědičnosti lze vytvářet libovolnou hierarchii layoutů, které lze mnohonásobně využít. Základem je definice bloku *{block}*, ten tvoří ohraničenou část HTML kódu, který může být v potomkovi přepsán či doplněn. [43]

Z pohledu rychlosti nemá šablonovací systém velký dopad na načítání stránky. Šablonovací engine kompiluje zdrojovou šablonu do nativního PHP. V prohlížeči se následně vykresluje již zkompilovaná verze. [43]

4.9.1 Nette

Komplexní objektově navržený framewok, který je v České republice velmi populární (jeho autorem je Čech). Používají ho například stránky csfd.cz, uloz.to nebo mf.cz. Využívá srozumitelnou a úspornou syntaxi společně s kvalitním ladícím (debugovacím) nástrojem Tracy. Obdobou *controllerů* jsou v Nette *presentery*. Nette obsahuje rozhraní knihoven, které je objektové a je postaveno na MVC architektuře. Je tedy dobré znát principy objektově orientovaného programování. [45]

K použití Nette je potřeba *sandbox* neboli kostra webové stránky, do které se vkládají stránky nové. Je to vlastně předpřipravená verze webových stránek sloužící pro začátek vývoje. [46]

Latte

Nette v sobě obsahuje šablonovací systém Latte, který dále rozšiřuje o vlastní syntaxi. Ta je využitelná pouze v Nette frameworku. Jinak lze Latte použít i samostatně. Nejsilnější vlastností Latte, stejně jako ostatních šablonovacích systémů, je možnost rozložit stránku na sadu menších šablon. Z těch se pak stránka zpětně poskládá a zkompiluje. Je tak možné na více místech použít jeden kód, tím nedochází k jeho opakování a vše se tak snadněji upravuje. [47]

4.9.2 Zend

Zend se drží typické MVC architektury s klasickým *controllerem* a velkým počtem pluginů. Používají ho velké společnosti, na tomto frameworku funguje například bbc.co.uk. Stejně jako

Nette podporuje široké spektrum databází. Naopak zde chybí integrovaný šablonovací systém, místo něj v základu používá soubory *.phtml*, které mohou obsahovat jak HTML, tak PHP. Integrace vhodného systému je tedy čistě na vývojáři. [48]

4.9.3 Smarty

Odděluje aplikační a prezenční logiku. Aplikační logika znamená zejména „výpočty“ v PHP, spojení s databází a celkově práci s daty. Naproti tomu prezenční logika představuje tu část, která je zobrazována uživateli. [49] [50]

Na rozdíl od Zend a Nette se nejedná o celý PHP framework. Je to pouze šablonovací systém napsaný v PHP. Jeho jednoduchost může být výhodou, například při realizaci vlastního redakčního systému. Je velice rychlý, nové kompilace se provedou pouze po změně obsahu. Snadno se vytvářejí nové funkce nebo modifikátory proměnných, ty umí upravit výstupy v požadovaném formátu, délce atp. Umí řídící struktury, obsahuje širokou škálu funkcí, modifikátorů a dobrou technickou podporu. [49] [51]

5 Vlastní zpracování

Tato kapitola obsahuje vlastní, tedy praktickou část této práce. Nejprve si představíme samotné zadání s jeho požadavky. Následovat bude úvod a zhodnocení jednotlivých výběrových kritérií, kterými může být práce realizována a bude proveden výběr.

Dalším krokem je grafická podoba webových stránek, kde si uvedeme podklady zaslané chovateli. Posléze proběhne tvorba a inicializace pracovního prostředí, včetně instalace komponent potřebných zejména pro kodérské práce.

Následovat budou kapitoly s vlastním systémovým řešením. Postupně budou popsány jednotlivé realizované prvky a na závěr optimalizační testy.

5.1 Souhrnné zadání

Webové stránky jsou tvořeny pro chovatele, který požaduje, aby mohl na stránky vkládat svou aktuální nabídku mastného plemenného skotu a společnou dokumentaci (výběrové protokoly, katalogy býků). V jeho nabídce jsou burské kozy a charolaiský skot. Jedinci, kteří nesplňují kritéria uchovnění nebo se z nějakého důvodu chovateli nelíbí, jsou poraženi a následně prodáni v chovatelově řeznictví. Toto řeznictví je v provozu pouze v době porážení zvířat.

Hlavním cílem je tedy prodej. Z tohoto důvodu jsou webové stránky z části pojaty formou internetového obchodu, kde jednotliví jedinci přestavují produkty. Samotný nákup je pak proveden po telefonické domluvě mimo toto řešení.

Chovateli byly navrženy další funkce mimo zadání, které mají za cíl zvýšit produktivitu a komfort oproti stávající situaci. Zadání je tak doplněno o registraci uživatelů do newsletteru, vyskakovací okno, kterým může návštěvníky informovat o otevření řeznictví, dále výpis již prodaných jedinců, zobrazení počtu nahraných fotografií u produktů, obsahová editace stránek v administraci, automatické vyplňování základních SEO atributů a systémovou přípravu na plynulý přechod na plnohodnotný internetový obchod.

Zvolené technologie i grafické zpracování bylo čistě na tvůrci diplomové práce, přičemž grafická podoba byla před samotnou realizací konzultována.

5.2 Výběr technologií a kritéria výběru

Před začátkem realizace proběhla volba kritérií, podle kterých byly vybrány optimální technologie pro vlastní realizaci. Mezi tyto kritéria patří:

- provozní cena,
- udržovatelnost,
- rozšiřitelnost,
- používatelnost.

Kritéria s nejvyšší váhou jsou používatelnost a provozní cena, následuje rozšiřitelnost a udržovatelnost. Je tedy vhodné vybrat takové technologie, které nevyžadují velké finanční náklady na svůj běh a zároveň jsou dobře a snadno používatelné.

Aby byl splněn požadavek na nízkou provozní cenu, byl vybrán linuxový server, který je oproti řešení od Microsoftu levnější. Jako programovací (skriptovací) jazyk bylo zvoleno PHP a databáze MySQL. Základna programátorů v PHP je široká, a tak i případné úpravy nebo správa třetí osobou bude snazší.

Kodérské práce byly realizovány v HTML a CSS za použití preprocesoru. Při výběru byl kladen důraz na jedno z dalších kritérií, a tím je vysoká používatelnost. Jako vhodné řešení se tak nabízel framework Bootstrap 3. Ten obsahuje sadu předpřipravených nástrojů a jistou míru unifikace, která je u administračního rozhraní přínosná. U třetí verze tohoto frameworku je ideální spojení s preprocesorem LESS.

Při tvorbě administrace byla použita šablona AdminLTE 2, která je dostupná zdarma a obsahuje řadu UI komponent. V kombinaci s Bootstrap 3, tak vzniklo graficky předpřipravené prostředí. Některé jeho části byly upraveny podle našich potřeb, jinak byla snaha o co nemenší úpravy, abychom zachovali znovupoužitelnost komponent.

5.3 Grafický návrh

Vizuální podoba webových stránek vychází z dvoubarevného schématu, kdy primární barvu představuje barva kontrastní, sekundární je pak barva bílá, která slouží jako podkladová. Zvolená primární barva je barva červeno-hnědá s HEX zápisem #903a22, ta je získána z předlohy loga viz Obr. 5.1, které sloužilo jako podklad pro tvorbu jeho vektorizované verze. Tato barva byla dále využívána pro konverzní prvky. Konverzními prvky jsou pro nás primárně tlačítka, hlavním je tlačítko pro volání, které se nachází v detailu produktu.

Na hlavní stránce jsou veškerá primární tlačítka horizontálně centrována v logické posloupnosti, což v kombinaci s navrženým minimalistickým *flat designem* nevyvolává potřebu unikátní konverzní barvy.

Obr. 5.1 Předloha loga pro vektorizaci

Vektorizovaná verze (viz Obr. 5.2) byla použita nejen jako centrální grafický prvek uprostřed navigaci, ale také pro oddělovače.

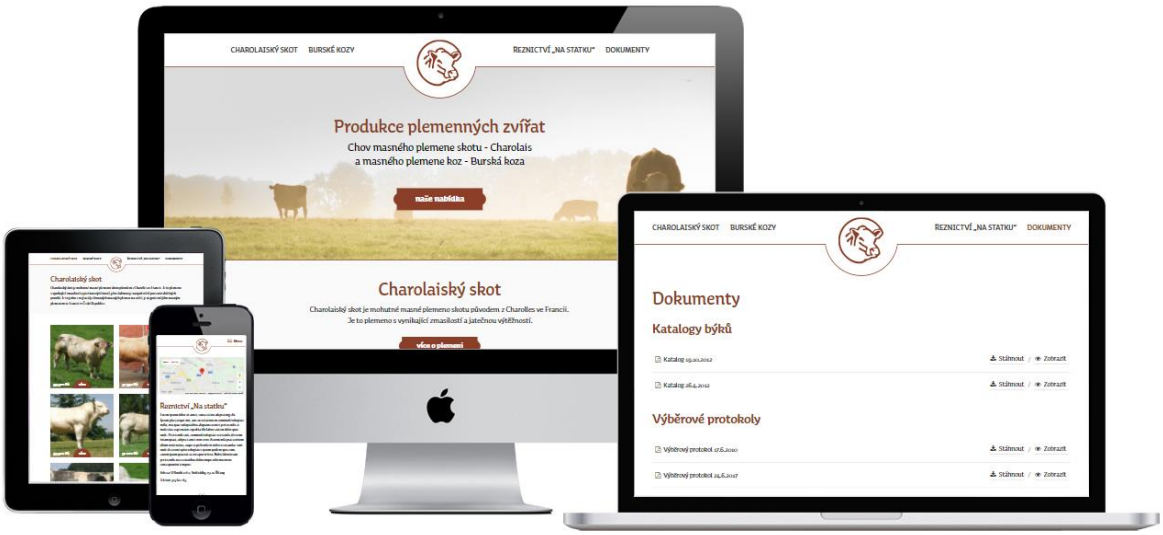
Obr. 5.2 Vektorizované logo

Vzhledem k nízkému počtu jedinců, kterými chovatel disponuje a vhodnosti použití větších náhledových obrázků pro zobrazení lepších detailů, bylo využito gridu 3 × N. Tedy při typickém rozlišení 1 920 px × 1 080 px jsou v řádku zobrazeny maximálně tři produkty o rozměrech 370 px × 370 px. To při velikosti kontejneru (kontejner definuje maximální možnou šířku stránky) 1 070 px tvoří jednu třetinu pro jeden produkt z celkové obsahové šířky stránky. Tento grid je pak v případě produktů použit napříč celými webovými stránkami.

Výstupem prvotního návrhu byl obrázek hlavní stránky viz příloha I a mockup viz příloha II neboli návrh zobrazený na reálných zařízeních (Apple iMac a Apple MacBook). Kodérské práce proběhly podle tohoto návrhu a další úpravy se již řešily v průběhu realizace. Jednotlivé stránky jsou také umístěny v přílohách viz Příloha III – Příloha VII spolu s administrační částí viz Příloha VIII. Na té je zobrazena úvodní stránka administrace produktů a jejich zkrácený výpis.

5.3.1 Responzivní verze

Součástí realizace je i plně responzivní verze viz Obr. 5.3, tedy verze přizpůsobená pro různá zařízení. Webové stránky se nevytvářely v desktopové a mobilní verzi, nýbrž byly využity vlastnosti CSS, kterými jsou *media queries*. Pomocí nich jsou stránky optimalizované pro všechny používané rozměry (rozlišení).



Obr. 5.3 Responzivní verze stránek

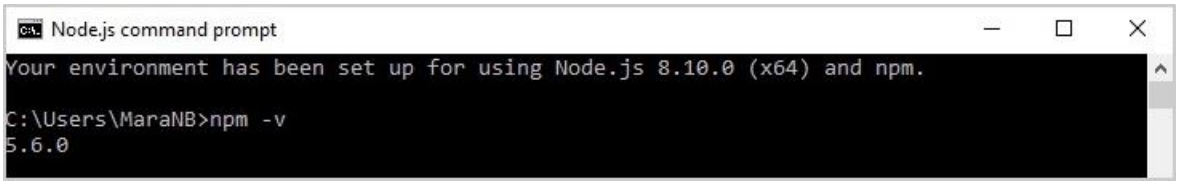
5.4 Pracovní prostředí

Kapitola je věnována instalaci, nastavení a struktuře lokálního pracovního prostředí na operačním systému Windows, potřebného pro efektivní tvorbu webové stránky zadaného rozsahu. Dále nastavení konfiguračních souborů tak, aby bylo prostředí co nevíce automatizované.

5.4.1 Node.js a NPM

Nejprve je potřeba provést instalaci frameworku Node.js, aby bylo možné na serveru spouštět JavaScripty pomocí příkazové řádky. Ze stránek nodejs.org byl stáhnut potřebný instalátor pro 64 bitový systém. Společně s jeho instalací proběhla také instalace Node Package Manager (správce JavaScriptových balíčků).

Test správnosti instalace je možné provést pomocí příkazové řádky Windows nebo Node.js viz Obr. 5.4.



Obr. 5.4 Zobrazení verze NPM v příkazové řádce

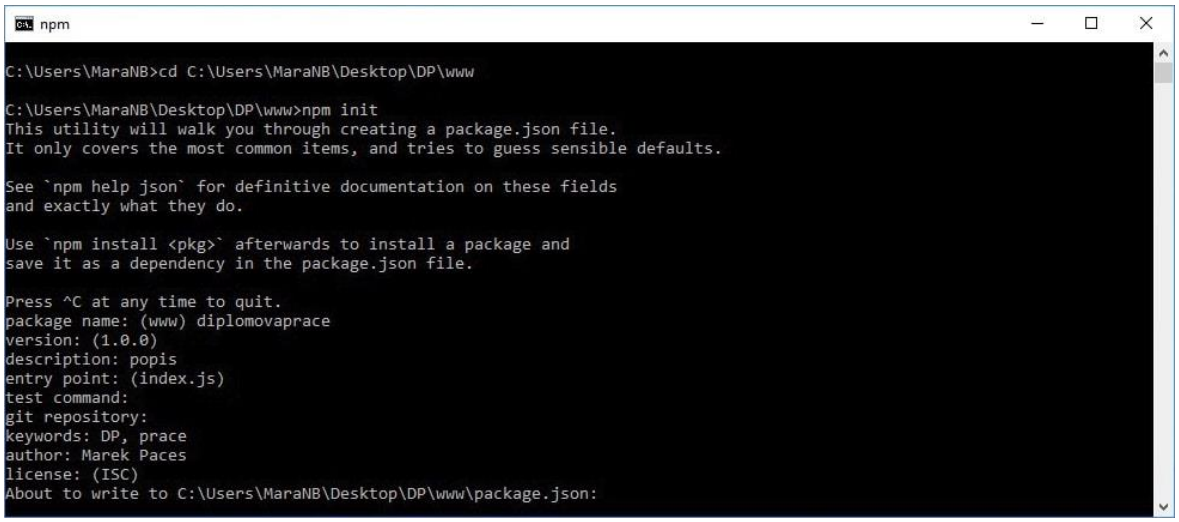
5.4.2 Grunt

Tím bylo připraveno prostředí pro sestavovač Grunt.

```
npm install -g grunt-cli
```

Instalace byla globální, na rozdíl od jednotlivých pluginů, které se instalují lokálně pro vlastní projekt. Následovala inicializace prostředí, nejprve vygenerování základního konfigurační souboru *package.json* viz Obr. 5.5. To již proběhlo v pracovní složce pro kódování webových stránek.

```
npm init
```



Obr. 5.5 Grunt instalace

Do vygenerovaného souboru byly zapsány požadované závislosti (*devDependencies*). Výsledný konfigurační soubor má pak následující tvar.

```
{
  "name": "diplomovaprace",
  "version": "1.0.0",
  "description": "popis projektu diplomove prace",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "dp",
    "prace"
  ],
  "author": "Marek Paces",
  "license": "ISC",
  "devDependencies": {
    "browser-sync": "^2.18.13",
    "grunt": "~0.4.5",
    "grunt-contrib-less": "^1.4.1",
    "grunt-contrib-watch": "^1.0.0",
    "grunt-inject-css": "^0.1.8",
    "gulp": "^3.9.1",
    "jit-grunt": "^0.10.0"
  },
  "dependencies": {
    "lesshat": "^4.1.0"
  }
}
```

Spuštění instalace vytvořilo adresář *node_modules*, do kterého se stáhly balíčky z definovaných závislostí.

```
npm install –only=dev
```

Základ byl tak hotov a následovala instalace dalších balíčků pomocí příkazů.

```
npm install jit-grunt -g
npm install grunt-browser-sync
npm install grunt-inject-css –save-dev
npm install browser-sync bs-html-injector
```

V dalším kroku byl vytvořen vedle *package.json* konfigurační soubor *Gruntfile.js*, do kterého se umisťují samotné úlohy. Vzhledem ke zvolenému preprocesoru LESS, bylo vhodné vytvořit úlohu, která umí zautomatizovat celý kompilační proces.

```
less: {
      development: {
        options: {
          compress: false,
          yuicompress: false,
          optimization: 0
        },
        files: {
          "css/style.css": "_source/less/style.less // cíl/destinace
        }
      }
    }
```

Aby kompilátor věděl, kdy má začít pracovat, byla vytvořena úloha, která sleduje provedené změny. Tedy ve chvíli, kdy je soubor uložen ve změněné verzi, se zároveň vyšle impuls ke kompilaci.

```
watch: {
      styles: {
        files: ['_source/less/**/*.less'], // jaký soubor sledovat
        tasks: ['less'],
        options: {
          nospawn: true
        }
      }
    }
```

Z pohledu automatizace tak bylo vyřešeno kompilování, následovalo nastavení úlohy, která má opět za úkol sledování změn, jen se nyní jedná o soubory s koncovkou *.tpl*. Tyto soubory

obsahují převážně HTML kód, který se má po změně automaticky aktualizovat v prohlížeči, ideálně bez potřeby obnovení stránky.

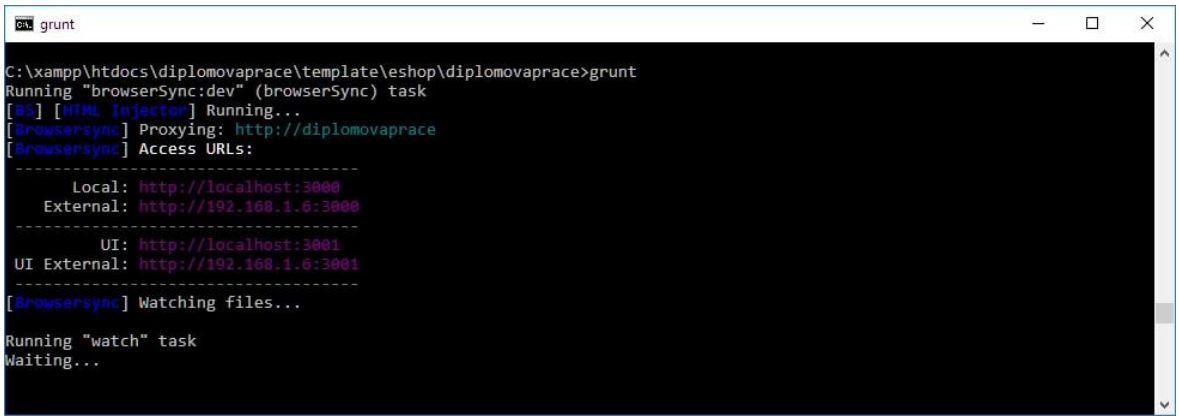
To samé platí i pro již zkompilovaný LESS kód, který byl převeden do souboru *.css*. Bylo nutné nakonfigurovat úlohu tak, aby sledovala provedené změny a tyto změny promítla v prohlížeči.

```
browserSync: {
        dev: {
            bsFiles: {
                src: [
                    'css/*.css',
                    'tpl/**/*.tpl'
                ]
            },
            options: {
                watchTask: true,
                proxy: "diplomovaprace",
                plugins: [{
                    module: "bs-html-injector",
                    options: {
                        files: "tpl/**/*.tpl",
                    }
                }]
            }
        }
    },
    bsReload: {
        css: "main.css"
    }
```

Na konci se provede samotné spuštění. Tím je konfigurační soubor připravený.

```
  grunt.loadNpmTasks('grunt-contrib-less');
  grunt.loadNpmTasks('grunt-browser-sync');
  // definice defaultní úlohy
  grunt.registerTask('default', ['browserSync', 'watch']);
```

Spuštění příkazu *grunt* (viz Obr. 5.6) z příkazové řádky spustí úlohu *browserSync* a automaticky se otevře okno prohlížeče s lokální URL adresou, která se schovává pod námi definovanou proxy adresou lokálního serveru.



Obr. 5.6 Spuštění úlohy browserSync

Aby bylo možné na webové stránky lokálně přistupovat, bylo potřeba mimo instalace lokálního serveru také vytvořit *vhost* (virtuální host), který nasměruje URL adresu na lokální souborové uložiště.

```
<VirtualHost diplomovaprace:80>
    DocumentRoot "C:\xampp\htdocs\diplomovaprace"
    ServerName diplomovaprace
    ServerAlias diplomovaprace
</VirtualHost>
```

Tato adresa se společně s adresou lokálního serveru zapíše do souboru *hosts*, tím dojde k nasměrování lokální adresy *127.0.0.1* na adresu *diplomovaprace*.

Cesta k souboru hosts je následující:

```
\Windows\System32\drivers\etc
```

Zapsaný řádek je ve tvaru:

```
127.0.0.1 diplomovaprace
```

Služby lokálního serveru poskytuje program XAMPP, který zaštiťuje prostředí pro běh PHP, Apache a MySQL.

5.5 Systémové řešení

Bylo využito služeb šablonovacího systému Smarty, rozšířeného do systémového řešení pro internetový obchod. Vytvořili jsme modulární prostředí, které bude podporovat budoucí vývoj. Dále administrační rozhraní pro správu kategorií a produktů, moduly pro registraci návštěvníkových e-mailů a telefonních čísel (newsletter), vyskakovací okno pro nově příchozí a modul se správou dokumentů. Při programování byl kladen důraz na aplikaci objektového přístupu.

Nyní je vhodné si přestavit struktury realizovaného systému a šablon samotných. Ilustrace schopností Smarty a uvedení příkladů z naší aplikace.

5.5.1 Struktura systému a šablon

Šablonovací systémem Smarty je využíván napříč celým systémem**.** Podíváme se na některé praktické ukázky jeho aplikace, díky kterým lépe pochopíme strukturu našeho systému. Začneme s tím, jak pomocí tohoto systému řešíme skládání zdrojových souborů *.tpl* do sebe.

Hlavní souborem je vždy *layout.tpl*, do kterého jsou vkládány obsahy ostatních souborů šablon, až poté dojde ke kompilaci do HTML. Obsahuje tu část kódu, kterou mají všechny stránky společnou. Jedná se tedy o hlavičku dokumentu s odkazy na CSS soubory, JavaScripty, dynamicky generovaný titulek, meta tagy pro sociální sítě a další.

V tomto layout jsou definovány tzv. bloky pomocí tagu *{block}*. Obsah těchto bloků je přepisován či doplňován jednotlivými šablonami, které jsou právě volány. Dále obsahuje hlavní navigaci (menu stránky) a blok *{content}*, do kterého je vkládán obsah požadované stránky. Mezi společné prvky patří i patička a globální JavaScripty umístěné v bloku *{scripts}*, který lze dále rozšiřovat.

Takto vypadá velice zjednodušená kostra, která se nachází v *layout.tpl*.

```
<!DOCTYPE html>
<html>
<head>
    {block head}{/block}
    <title>
        {block title}
            {$smarty.block.child}{System\Config::Get("TITLE")}
        {/block}
    </title>
</head>
<body>
    <nav>
        <!-- globální navigace -->
    </nav>

    <!-- prostor pro obsah -->
    {block content} {/block}

    <footer>
        <!-- globální patička -->
    </footer>

    {block scripts}
        <!-- globální JavaScripty -->
    {/block}
</body>
</html>
```

Na konkrétní stránce se volá pomocí PHP skriptu zobrazení koncové TPL šablony, tedy např. *kontakt.tpl*. Konkrétní šablona následně rozšiřuje hlavní šablonu *layout.tpl* a nahrazuje vybrané bloky požadovaným obsahem. Kód v *kontakt.tpl* by tedy vypadal následovně:

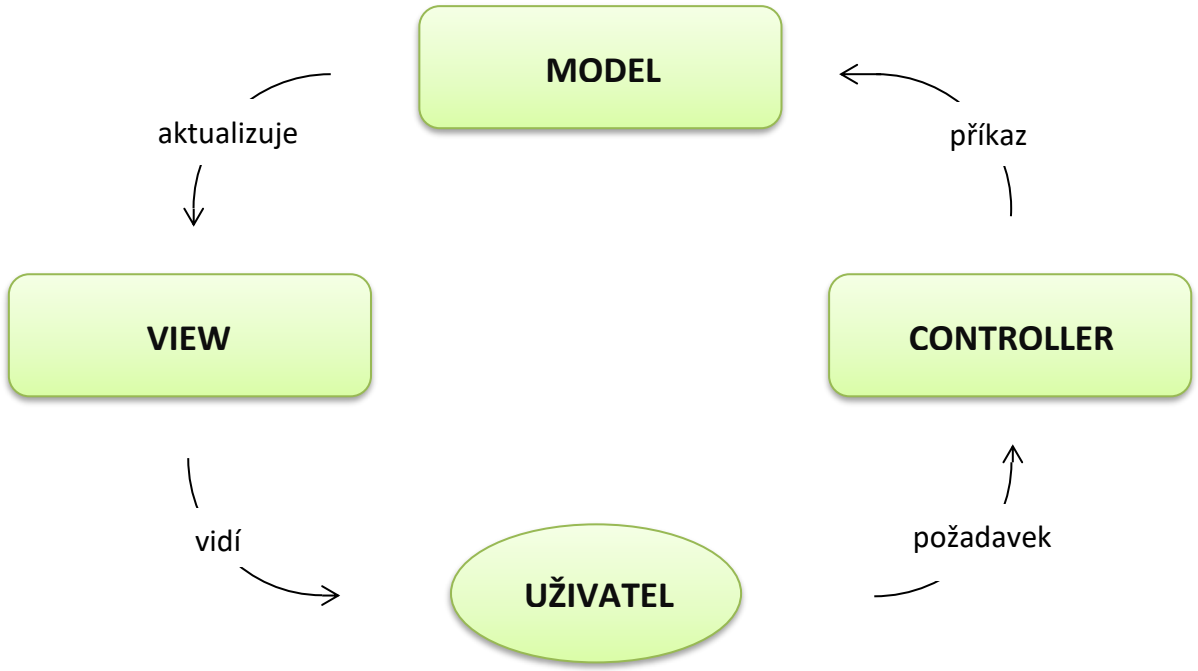
```
{extends file="layout.tpl"}
{block title}Kontakt{/block}

{block content}
     <!-- obsah stránky Kontakt -->
{/block}

{block scripts}
     <!-- doplňující JavaScripty pro stránku Kontakt -->
{/block}
```

Tímto způsobem se do sebe skládají šablony, které tvoří výslednou souborovou strukturu pro frontend i backend. Zjednodušený adresářový strom celého systému má následující podobu:

```
├── data
|   ├── app
|   |   ├── bin
|   |   ├── config
|   |   ├── module
|   |   └── office
|   ├── bin
|   |   ├── extension
|   |   ├── smarty
|   |   └── system
|   └── tmp
├── image
└── template
    ├── dashboard
    └── eshop
```

Adresáře *dashboard* a *eshop* ve složce *template* představují právě to místo, kde se s šablonami pracuje především. Adresář *dashboard* obsahuje administraci a v *eshop* je frontendové řešení. Právě v těchto místech byla vytvořena pracovní prostředí pro kodérské práce.

Aplikační logika je v systému od šablon oddělena. Hlavní *model* je realizován více objekty, které společně tvoří jádro systému, sloužící ke zpracování dat. *Model* dále aktualizuje pohled na příkaz řadiče. Samotná komunikace včetně uživatele (viz Obr. 5.7) v této systémové architektuře vypadá následovně. [52]



Obr. 5.7 MVC model včetně uživatele [52]

Vytvořený systém využívá globální *model*, který je společný jak pro frontend, tak pro backend. Cesta k němu je následující.

```
\data\bin
```

Adresář bin obsahuje *extension*, *smarty* a *system*. Ve *smarty* se nachází samotný šablonovací systém, v *extension* různá rozšíření (sociální sítě, skloňování atp.) a *system* představuje systémové jádro aplikace.

Pokud chceme vytvořit *model* lokální, například pouze pro administraci, učiníme tak v následující složce.

```
\data\app\office\dashboard\model
```

Zde se řeší funkční části, které jsou potřeba pouze pro administraci. Častěji však programování zasahuje do frontendu i backendu, a to z důvodu tvorby administrace pro realizovaný modul. V tom případě pracujeme v tomto adresáři.

```
\data\app\bin
```

Zde se nacházejí *modely* například pro produkty, řešení správy dokumentů na stránkách, newsletter nebo pro vyskakovací okno.

Celý systém podporuje modulární tvorbu, lze tak realizovat jednotlivé moduly s mikro MVC architekturou, které je možno kdykoli doinstalovat a dále rozšiřovat. Tyto modulární prvky se nacházejí v adresáři module.

```
\data\app\module
```

Do budoucna lze rozšířit systém o nové funkce, které budou od chovatele požadovány a nebude nutný zásah do systémového jádra.

Controller má v systému na starosti tok událostí a aktualizuje samotný *model*, stará se tedy o funkční provázání. Konkrétně koordinuje komunikaci mezi *modelem* a *view*. [52]

Následující umístění *controlleru* patří řešení detailu produktu. Opět je rozlišován backend od frontendu, kdy adresář *eshop* reprezentuje frontend, do něhož detail produktu patří.

```
\data\app\office\eshop\page\product
```

5.5.2 Prezentace dat pomocí Smarty

Po seznámení se strukturou šablon a systému se nyní podíváme na to, jakým způsobem nám Smarty pomáhá s prezentací dat v šablonách. Budou vysvětleny základní řídící struktury a cykly. Těmi jsou například *if*, *else* nebo *foreach* a speciální vlastnosti. Ty mohou být přidány do cyklů a využívány například pro výpis dokumentů. Následně se struktury aplikují ve vlastním řešení.

Používané podmínky jsou *{if}*, *{elseif}* a *{else}*. U podmínek operátory *==*, *<*, *>*, *>=*, *<=*, *!=*, *===* a *!==*. Dále standartní operátory z PHP *and* a *or* (*&&, ||*). Následuje tabulka viz Tab. 5.1, ve které jsou vypsány všechny dostupné operátory, jejich alternativní zápis, příklad zápisu, význam a ekvivalent v PHP. [53]

Tab. 5.1 Přehled operátorů [53]

Operátor	Alternativa zápisu	Příklad zápisu	Význam	PHP ekvivalent
==	eq	$a eq $b	equals	==
!=	ne, neq	$a neq $b	not equals	!=
>	gt	$a gt $b	greater than	>
<	lt	$a lt $b	less than	<
>=	gte, ge	$a ge $b	greater than or equal	>=
<=	lte, le	$a le $b	less than or equal	<=
===		$a === 0	check for identity	===
!	not	not $a	negation (unary)	!
%	mod	$a mod $b	modulous	%

is [not] div by		$a is not div by 4	divisible by	$a % $b == 0
is [not] even		$a is not even	[not] an even number (unary)	$a % 2 == 0
is [not] even by		$a is not even by $b	grouping level [not] even	($a / $b) % 2 == 0
is [not] odd		$a is not odd	[not] an odd number (unary)	$a % 2 != 0
is [not] odd by		$a is not odd by $b	[not] an odd grouping	($a / $b) % 2 != 0

V šabloně to pak může vypadat následovně.

```
{if $name eq 'Jan'}
    Vítejte Jane.
{elseif $name eq 'Adam'}
    Vítejte Adame.
{else}
    Vítejte ostatní.
{/if}
```

Nejpoužívanějším cyklem pro výpis záznamů je cyklus *{foreach},* v něm lze používat speciální vlastnosti *@first* a *@last*. Vlastnost *@first* v sobě drží hodnotu jedné iterace. Naopak vlastnost *@last* rozpozná iteraci poslední. Tím lze při výpisu identifikovat první a poslední záznamy v cyklu. To se hodí například u výpisu tabulky, kdy je možno pomocí *@first* vypsat nadpis tabulky a první párový tag *<table>*. Následuje *{foreach}* s výpisem samotných dat do řádků a *@last* ukončí tabulku *</table>*. Případně *{foreachelse}* vypíše skutečnost, že v tabulce nejsou žádná data. Cyklus *{foreach}* je tvořen ze dvou povinných a dvou nepovinných atributů viz Tab. 5.2.

Tab. 5.2 Atributy cyklu foreach

Atribut	Typ	Povinný	Popis
from	pole	Ano	Procházené pole
item	řetězec	Ano	Název proměnné v cyklu
key	řetězec	Ne	Index aktuální hodnoty
name	řetězec	Ne	Unikátní pojmenování cyklu

Opět následuje příklad možného použití v šabloně.

```
{foreach from=$promenna item=vypis key=index name=nazev}
     Tělo cylku
{foreachelse}
     Proměnná je prázná
{/foreachelse}
```

Takto lze obecně pracovat se Smarty v šablonách. Následují dvě praktické aplikace, které jsou použity v řešení. Nejprve přidávání třídy *active* položce v navigaci, která odpovídá stránce, na které se návštěvník nachází.

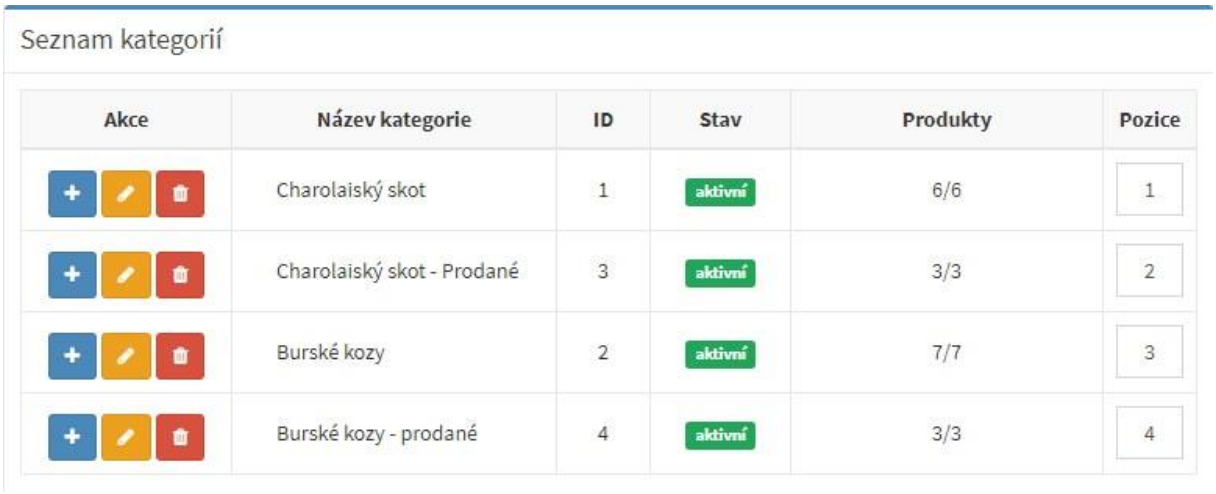
```
<li
{if $smarty.server.REQUEST_URI == '/dokumenty'} class="active"{/if}>
     <a href="{link href="/dokumenty"}">Dokumenty</a>
</li>
```

Podle identifikátoru kategorie je možné v detailu produktu vypsat prodané produkty, které se nacházejí v samostatné kategorii. Takto vypadá výpis jednoho řádku, neboli tří produkty.

```
{if $category->getId() == 1}
{$category = Eshop\Category\Category::Get(3)}
{foreach $category->getProducts(false,3) as $product}
<div class="produkt">
     <!-- prodaný produkt -->
</div>
{/foreach}
{/if}
```

5.5.3 Kategorie a produkty

Výpis a kategorizace produktů je hlavní funkcí webových stránek. Rozlišujeme celkem čtyři kategorie viz Obr. 5.8, které chovatel požaduje. Na frontendu je přístup do kategorií *Charolaiský skot* a *Burské kozy* možný z hlavní stránky přes tlačítko „zobrazit celou nabídku“ nebo pomocí hlavní navigaci.

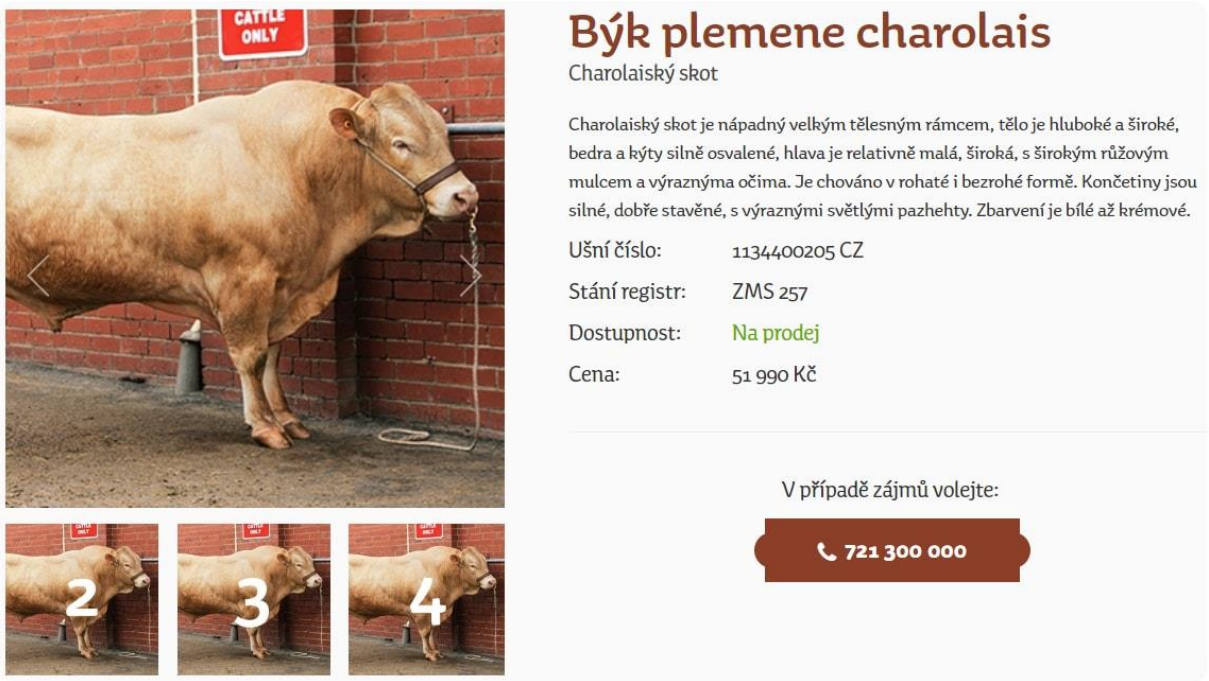


Obr. 5.8 Seznam kategorií

Chovatel může v administraci kategorie upravovat a měnit tak například jejich název, popis zobrazený v detailu kategorie nebo parametry SEO. Do kategorií lze zařazovat produkty, u kterých se vyplňují zejména následující atributy:

- název produktu,
- popis produktu,
- ušní číslo (unikátní identifikátor),
- státní registr,
- zařazení do kategorie,
- cena (nulová cena vypíše na frontendu „cena dohodou“),
- viditelnost produktu,
- dostupnost,
- skaldem celkem (podmínkou je alespoň jeden kus),
- hlavní a vedlejší obrázky.

Dále lze upravovat parametry SEO, které se předvyplňují automaticky a URL adresu (případně funkční aliasy této URL). Pokud dojde k prodání daného kusu (produktu), bude ručně zařazen do kategorie prodané. Následně se na stránkách zobrazí v detailu produktu viz Obr. 5.9, přičemž u produktů charolaiského skotu se nebudou zobrazovat burské kozy a opačně.



Obr. 5.9 Detail produktu

V realizaci bylo vyžadováno, aby zákazníci v případě zájmu chovateli zavolali, z tohoto důvodu je *call-to-action* prvkem tlačítko pro volání.

V souboru šablony, kde se řeší výpis produktů z kategorie je potřeba rozšířit základní *layout.tpl*. Poté se zapíše dynamicky generovaný titulek (*title*) a do obsahového bloku *{block content}* se vloží obsah stránky.

```
{extends file="include/layout.tpl"}
{if $PAGE->title}{block title}{$PAGE->title} | {/block}{/if}
{block content}
<div id="category-header">
    <div class="container">
        <h1 class="heading">{$category->getName()}</h1>
        {if $category->getDescription()}
        <div class="description">
            {$category->getDescription()}
        </div>
        {/if}
    </div>
</div>
<div id="category-products">
    <div class="container no-padding">
        {include file="category/products.tpl"}
    </div>
</div>
{/block}
```

Obsahová část stránky se skládá ze jména kategorie, popisu kategorie (pokud je vyplněný v administraci) a samotného výpisu produktů. Ten je vkládán z další (vnořené) šablony s následujícím kódem.

```
<div class="container">
    <div class="products">
        <div class="row">
            <div class="flex-container">
                {include file="category/product_list.tpl"}
            </div>
        </div>
    </div>
</div>
```

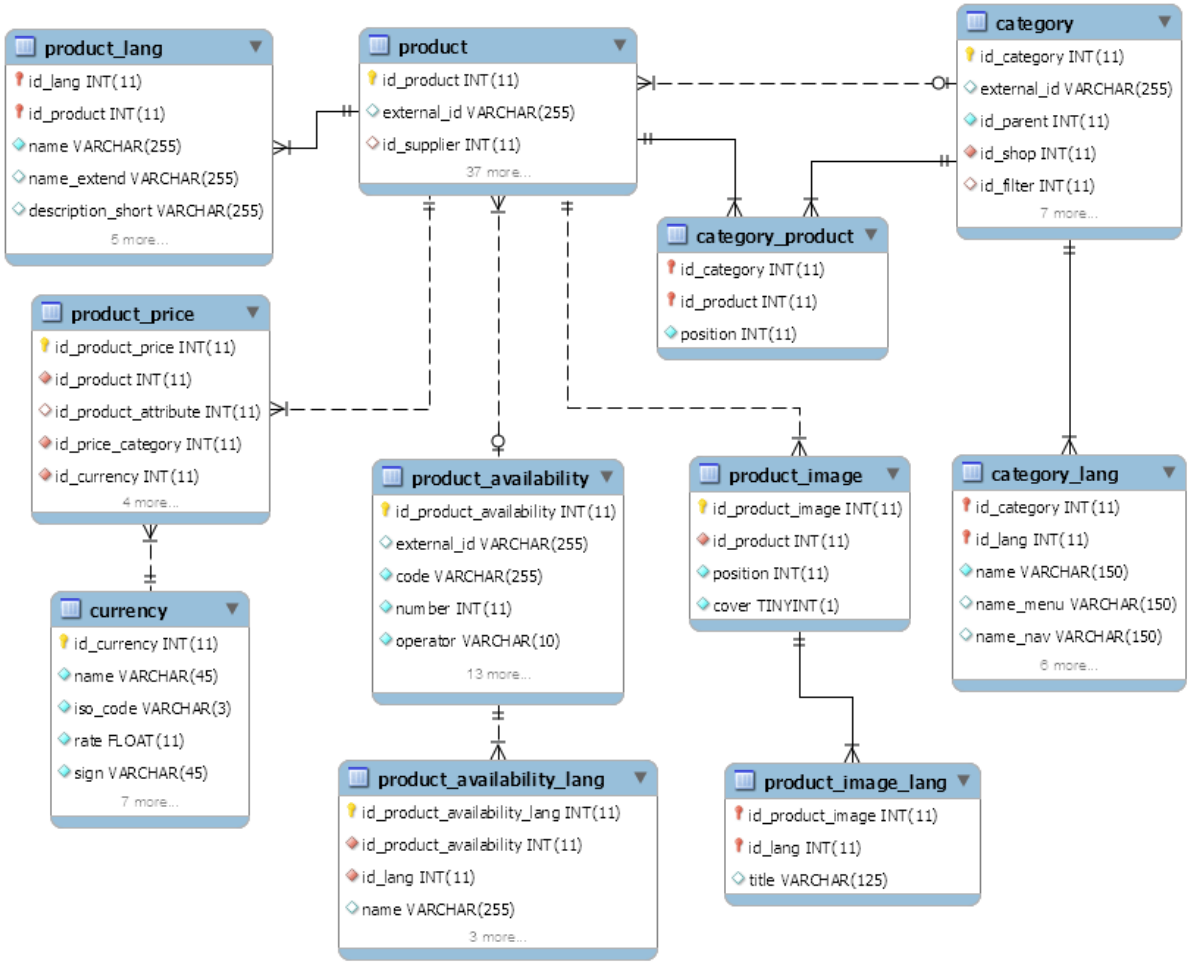
V této šabloně se vypíší konečné produkty ze souboru *product_list.tpl*. Tyto produkty se nachází uvnitř *flex-container*. Je tedy patrné, že je zde použito pružné stylování CSS Flexbox.

V souboru *product_list.tpl* se nachází zmíněný cyklus *{foreach}*, pomocí něj se vypíší všechny produkty z kategorie, které splňují podmínky pro vypsání definované v systému.

```
{foreach $products as $product}
<div class="flex-item product-item">
     <a href="{link href=" /product/detail/{$product->getLink()}"}"
class="inner">
        <img src="{$product->getCover()->getImage("item")}"
          alt="{$product->getName()}" title="{$product->getName()}">
        <button href="#" class="btn-detail">více</button>
        <span class="price">
           {if $product->getPrice()=="0 Kč"}
                dohodou
                 {else}
                {$product->getPrice()}
           {/if}
     </span>
     {if count($product->getImages()) > 1}
          <span class="images"><i class="fa fa-camera"></i>
          {count($product->getImages())}x</span>
     {/if}
    </a>
</div>
{/foreach}
```

Postupně se tak vypíše odkaz na produkt, hlavní obrázek, název, cena (nulová cena vypíše text „dohodou“) a počet obrázků (pokud je jich více jak jeden).

Po uvedení způsobu, jakým se produkty vypisují, následuje seznámení se systémem, který data pro výpis připravuje. Takto vypadá zjednodušený návrh části databáze, která řeší problematiku kategorií a produktů viz Obr. 5.10.



Obr. 5.10 Databázový model kategorií a produktů

Systém byl navržen tak, aby podporoval více jazyků. Proto jsou zde tabulky s přívlastkem *_lang*, které obsahují záznamy pro různé jazykové mutace.

Důležité vazby vypadají následovně. Jedna kategorie může mít více produktů, přičemž jeden produkt může mít více jazykových mutací. Produkty jsou dále přiřazovány do kategorií v tabulce *category_product*, kdy jeden produkt může být pouze v jedné kategorii. Ostatní tabulky řeší cenu produktu, jeho dostupnost a produktové obrázky.

Dalším krokem po návrhu databázového *modelu* je samotná realizace v PHP. Hlavní jsou *modely Category.php* a *Product.php*. Ty obsahují všechny potřebné proměnné a funkce. Každý *model* má svůj *controller* s názvem *detail.page.php,* nacházející se v příslušných složkách.

```
\data\app\office\eshop\page\category\detail.page.php
\data\app\office\eshop\page\product\detail.page.php
```

Vzhledem ke komplexnosti obou *modelů* bude uvedeno po jednom příkladu z každého. V *modelu* produktu to je funkce zařazující produkt do kategorie.

```
public function setCategory($id_category){
    if(DB::getInstance()->Exists("category_product",array(
            "id_category" => $id_category,
            "id_product" => $this->id
    )))
        return $this->setParentCategory($id_category);

    $insert = DB::getInstance()->Insert("category_product",array(
            "id_category"   => $id_category,
            "id_product"    => $this->id,
            "position"      => 0
    ));
    if($insert !== false){
        return $this->setParentCategory($id_category);
    }
}
```

Funkce předává jako parametr identifikátor kategorie, do které se má produkt zařadit. Zprvu proběhne kontrola, zdali již není produkt v kategorii zařazený. Kontrola probíhá podle identifikátorů kategorie a produktu. Pokud nastane situace, kdy již produkt do kategorie zařazen byl, proběhne pokus o zařazení do kategorie nadřazené (rekurzivně). V případě, že produkt zařazen ještě nebyl, potom se produkt zařadí. Následně se zařadí i do kategorie nadřazené (rodičovské).

Z *modelu* kategorie je uveden následující kód. Ten má za úkol vrátit seznam produktů v kategorii. Jako parametr předává maximální počet produktů, které se mají vrátit.

```
public function getProducts($sort = false, $limit = 6){
    if($this->products === null){
        $data = DB::getInstance()->SelectMultiQuery("
                SELECT product.id_product FROM product
                JOIN category_product ON
category_product.id_product = product.id_product
                WHERE
                        category_product.id_category = :id_category
                    AND product.deleted = :f
                GROUP BY product.id_product
                LIMIT $limit
            ",array(
                "id_category"   => $this->id,
                "f"             => 0
        ));
        $ret = array();
        if($data and is_array($data)){
            foreach($data as $v){
                $ret[] = new Product($v['id_product']);
            }
        }
        $this->products = $ret;
    }
    return $this->products;
}
```

Na začátku, funkce zkontroluje načtení produktů. Pokud ještě nebyly načteny, tak se načtou z databáze. Pokud se data načetla, vytvoří se instance produktů. Seznam produktů se poté zapíše do proměnné a pomocí *return* se vrátí data s načtenými produkty.

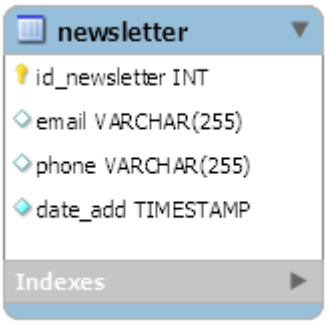
5.5.4 Realizace newsletteru

Součástí webový stránek je prezentace řeznictví, které se otevírá pouze v době porážek, to jest zhruba sedmkrát ročně. Chovatel kontaktuje zákazníky telefonicky. Nově tak dostává možnost kontaktovat je prostřednictví e-mailu, kdy se jak ti stálí, tak ti noví mohou přihlásit k odběru novinek viz Obr. 5.11.

Obr. 5.11 Frontendová podoba odběru novinek

Seznam e-mailů a telefonních čísel lze spravovat v administraci. Jelikož budou data uchovávána v databázi, bylo vše připraveno ke splnění obecného nařízení o ochraně osobních údajů (GDPR). Byl umístěn text se souhlasem včetně odkazu na systémovou stránku, na které se nachází předpřipravený text o zpracování osobních údajů. Tento text má chovatel možnost upravit (doplnit) v administraci.

Prvním krokem je vždy návrh databázového *modelu*. Modulu stačí jedna tabulka viz Obr. 5.12. Zapisuje se zákazníkův *e-mail*, nebo *telefon*, *datum a čas* (časová známka) kdy se registroval.



Obr. 5.12 Tabulka newsletter

Tabulka se skládá z následujících atributů (sloupců) viz Tab. 5.3.

Tab. 5.3 Popis tabulky z newsletteru

Id_newsletter (integer)	identifikátor záznamu v tabulce pro případnou editaci
Email (varchar)	e-mail, který zadává návštěvník do formuláře
Phone (varchar)	telefon, který zadává návštěvník do formuláře
Date_add (timestamp)	časová známka vložení záznamu do tabulky

Takto vypadá případ, ve kterém návštěvník klikne na tlačítko a vyvolá tak událost. Následuje asynchronní zpracování pomocí technologie AJAX, konkrétně v objektu newsletter.

```
$(document).on("click","#newsletter_btn",function(e){
   e.preventDefault();
   var str = $("#newsletter_input").val();
   newsletter.add(str, function(obj){
      if(obj && obj.status === 1){
          // V pořádku
          $('#modal-newsletter-success').modal('show');
      }else{
          // Chyba
          $('#modal-newsletter-unsuccess').modal('show');
      }
   });
   return false;
});
```

Informace o stavu odeslání jsou návštěvníkovi prezentovány prostřednictvím modálního dialogového okna (modal), který je komponentou frameworku Bootstrap. Objekt newsletter pak vypadá následovně.

```
var newsletter = {
    add: function(str, callback){
        $.ajax({
            method: "POST",
            url: "/ajax/newsletter/add",
            data: {str: str},
            cache: false
        }).done(
            function(res){
                var obj = $.parseJSON(res.trim());
                if(obj){
                    callback(obj);
```

```
                }else{
                    callback(false);
                }
            }).fail(function(jqXHR, err){
            console.log("Request failed: " + err);
        });
    }
};
```

Nyní již přichází na řadu samotný *model Newsletter.ph*p. Ten je tvořen objektem s veřejnými funkcemi řešícími přidávání, odstraňování a výpis. Hlavní metodou objektu je tzv. konstruktor *__construct*. Ten při volání nové instance objektu nastaví patřičné hodnoty objektu dle vstupních dat. Případně, pokud je potřeba, načte potřebná data z databáze. K tomu využívá objekt *DB.class.php*, který je umístěn v jádru systému.

```
public function __construct($id = null, $email = null, $phone =
null, $date_add = null) {
    if($id and $date_add){
        $this->id = $id;
        $this->email = $email;
        $this->phone = $phone;
        $this->date_add = $date_add;
    }elseif($id){
    $data = DB::getInstance()-
>Select("newsletter",["id_newsletter"=>$id]);
        if($data and isset($data['id_newsletter'])){
            $this->id = $data['id_newsletter'];
            $this->email = $data['email'];
            $this->phone = $data['phone'];
            $this->date_add = $data['date_add'];
        }
    }
}
```

Controllerem je *newsletter.page.php*. Ten dědí metody z rodičovského objektu *page.class.php*. Obsahuje dvě hlavní funkce: *load* a *loadAjax*, které jsou volány systémem v závislosti na typu příchozího požadavku od návštěvníka. V našem případě se jedná o požadavek, který je zpracováván na pozadí stránky. Systém tedy volá metodu *controlleru loadAjax.*

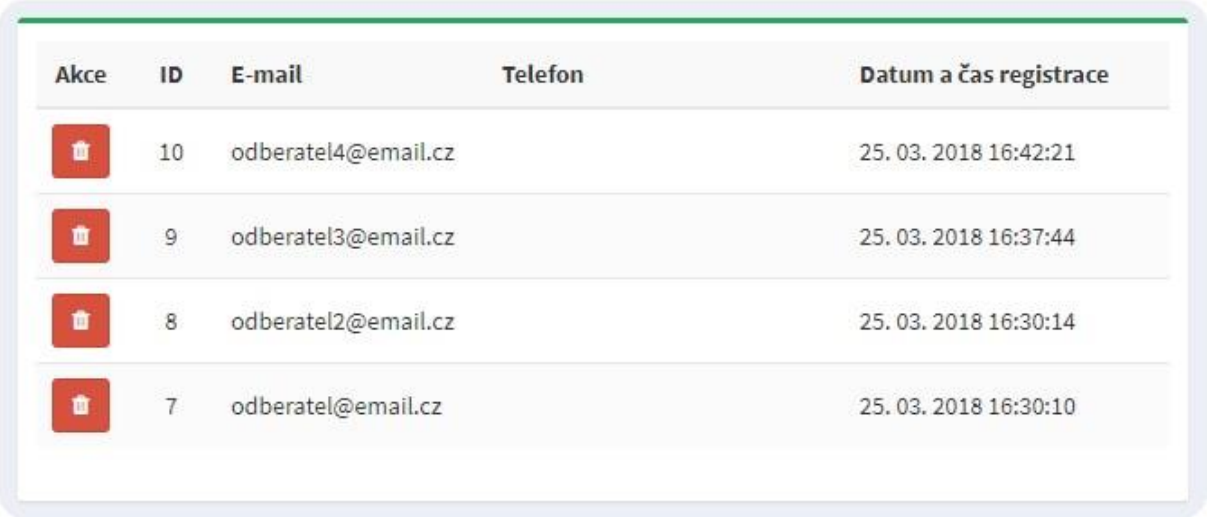
```
public function loadAjax(&$params){
    if(in_array("add",$params)){
        $this->addNewsletter();
    }
    return;
}
```

A pomocí své privátní funkce rozhodne o stavu odeslání.

```
private function addNewsletter(){
    if(Newsletter::add(Post::Get("str"))){
        echo json_encode([
            "status" => 1
        ]);
    }else{
        echo json_encode([
            "status" => 0
        ]);
    }
    return false;
}
```

Jelikož se jedná o přenos na pozadí bez grafického výstupu, tak výstupem funkce není volání šablony, ale řetězec ve formátu JSON, který přenáší informaci, zda vše proběhlo v pořádku, či nikoliv.

Administrační rozhraní (viz Obr. 5.13) pak vypisuje seznam e-mailů a telefonů včetně časové známky. Časová známa uchovává hodnotu odpovídající času zapsání záznamu. Nutnou funkcí u tohoto modulu je mazání záznamů kvůli normě GDPR.



Akce	ID	E-mail	Telefon	Datum a čas registrace
	10	odberatel4@email.cz		25. 03. 2018 16:42:21
	9	odberatel3@email.cz		25. 03. 2018 16:37:44
	8	odberatel2@email.cz		25. 03. 2018 16:30:14
	7	odberatel@email.cz		25. 03. 2018 16:30:10

Obr. 5.13 Administrační rozhraní newsletteru

Na tvorbu administračního rozhraní byl zapotřebí následující soubor. Na jeho začátku se nachází Smarty kódy pro rozšíření základní šablony a vložení šablony s notifikacemi.

```
{extends file="include/layout.tpl"}
{include file="include/statusbar.tpl"}
```

Náplň obsahem je realizována přes cyklus *foreach.* Ten vypíšeme řádky záznamů do bloku *content*.

```
{block content}
{foreach Newsletter\Newsletter::getList() as $n}
<tr>
    <td class="text-center">
        <a href="#delete" class="btn btn-sm btn-danger" data-
delete="{$n->getId()}">
            <i class="fa fa-trash"></i>
        </a>
    </td>
    <td class="text-center">{$n->getId()}</td>
    <td>{$n->getEmail()}</td>
    <td>{$n->getPhone()}</td>
    <td>{date format="d. m. Y H:i:s" time=$n->getDateAdd()}</td>
</tr>
{foreachelse}
<tr>
    <td colspan="3" class="text-center">
      V seznamu nejsou žádní odběratelé...
    </td>
</tr>
{/foreach}
{/block}
```

Na konci je napojen soubor s JavaScriptem, který obsluhuje zobrazování modalového okna pro mazání záznamů.

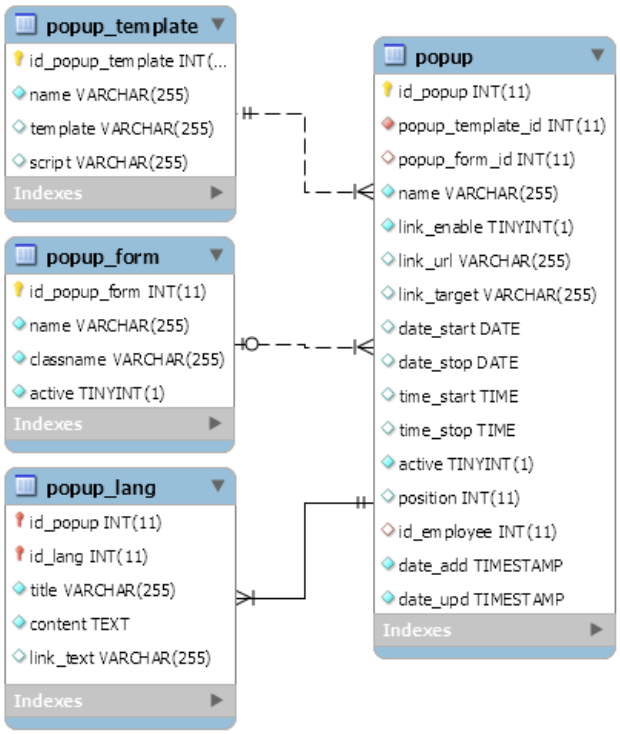
```
{block script append}
  <script src="{$TEMPLATE_URL}dist/js/pages/newsletter.js"></script>
{/block}
```

5.5.5 Realizace vyskakovacího okna

Zadání definuje požadavek na vytvoření vyskakovacího okna (pop-up) viz Obr. 5.14. Okno je určeno pro nově příchozí návštěvníky a má u něj být možnost nastavení data a času kdy se má návštěvníkům zobrazovat.

Obr. 5.14 Frontendová podoba vyskakovacího okna

Realizované okno obsahuje titulek, text a tlačítko, vše je napojené na administraci s možností editace. Modul využívá čtyři tabulky viz Obr. 5.15.



Obr. 5.15 Databázové tabulky patřící vyskakovacímu oknu

Atributy tabulky *popup* odpovídají funkcím, které požadujeme a rozšiřují je tabulkou *popup_form*, která řeší situaci, kdy je ve vyskakovacím okně navíc umístěn formulář (např. newsletter). Hlavní tabulka *popup* tak obsahuje *jedinečný identifikátor* (primární klíč), *jedinečný identifikátor šablony* tvořící cizí klíč (můžeme používat více šablon), *jedinečný identifikátor formuláře* tvořící cizí klíč, *jméno*, *atributy*

s prefixem *link_* pro tlačítko, *časový rozsah zobrazení*, *aktivitu*, *pozici*, *jedinečný identifikátor zaměstnance* (administrátora) tvořící cizí klíč, *datum přidání* a *poslední aktualizace*.

Tabulka *popup_template* je zde pro případ nasazení více vyskakovacích oken. Zároveň je v ní možnost zapsat název souboru se skriptem, který zajišťuje zobrazení tohoto okna. V *popup_lang* jsou zapsány ty texty, které se překládají pro další jazykové mutace.

Funkce pro nastavení obsahu je umístěna v *modelu PopUP.php* a má následující podobu.

```
public function setContent($content, $id_lang = false) {
    if($id_lang == false)
        $id_lang = Lang::getLang()->getId();
    $this->storage_translate[$id_lang]['content'] = $content;
    if($id_lang == Lang::getLang()->getId())
        $this->content = $content;

    return $this;
 }
```

Tato funkce nastaví obsah v jazyce, kterému odpovídá příslušný identifikátor. Pokud identifikátor chybí, pak se obsah nastaví dle právě používaného jazyka.

Vyskakovacích oken, resp. jejich šablon může být více. Samotný modul představuje mikro MVC architekturu. Ve *view* jsou umístěny základní šablony, které do sebe vnořují obsah podle zvoleného vyskakovacího okna.

```
\data\app\module\popup\view\tpl\blockmodals.tpl
\data\app\module\popup\view\tpl\blockfooterscript.tpl
```

Kód v nich má za úkol podle patřičného identifikátoru vypsat soubor s HTML kostrou a příslušný skript.

```
<!-- blockmodals -->
{if $POPUP and $POPUP->getId()}
    {include file=$POPUP_TPL}
{/if}

<!-- blockfooterscript -->
<script src="{$TEMPLATE_URL}js/jquery.cookie.js"></script>
{if $POPUP and $POPUP->getId()}
    {foreach $POPUP_SCRIPT as $src}
```

```
        <script src="{$TEMPLATE_URL}{$src}"></script>
    {/foreach}
{/if}
```

Kostra konkrétního vyskakovacího okna a skript se nachází ve složce patřící frontendu, tedy v *template**eshop*. Je tak zapotřebí funkce, která bude umět tyto soubory načíst.

Podle zadání se má okno zobrazovat pouze novým návštěvníkům. O to se stará následující skript. Tento skript vyžaduje API pro cookies, které není součástí používané knihovny jQuery.

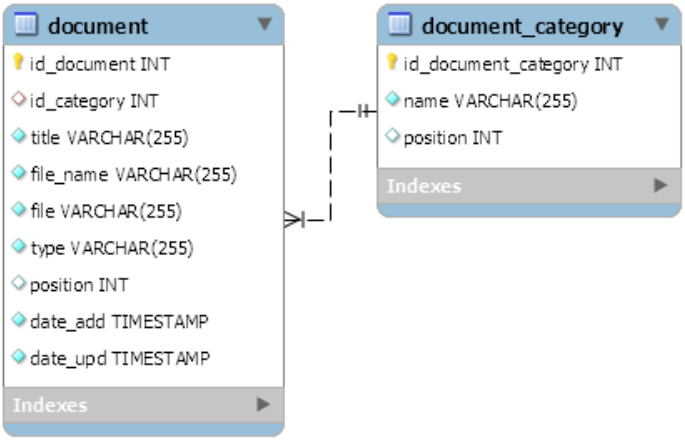
```
function showPopUP() {
    var pc = $.cookie("popup");
        if(!pc || parseInt(pc) != 1){
            $("#modal-popup").modal("show");
        $.cookie("popup", 1, {expires : 1});
    }
}
setTimeout("showPopUP();",5000);
```

Skript obsahuji funkci, která zkontroluje stav návštěvníkových cookies. Okno se zobrazí, pokud na stránkách ještě nebyl (nebo byl, ale časový interval od jeho poslední návštěvy je větší než jeden den).

Výsledná podoba administračního rozhraní viz Příloha X obsahuje vše co bylo na začátku navrženo v tabulkách databáze.

5.5.6 Realizace dokumentů

Posledním z realizovaných modulů byly *dokumenty*. Ty se na stránkách nacházejí zcela samostatně. Počítá se s nahráváním *katalogů býků* a *výběrových protokolů*. Z rezervních důvodů byla vytvořena ještě kategorie *ostatní*. Kategorie jsou tedy pevně dané. Bylo potřeba dvou tabulek viz Obr. 5.16. Jedna pro kategorizaci, druhá pro vlastní dokumenty.



Obr. 5.16 Databázové tabulky patřící dokumentům

V jedné kategorii může být více dokumentů. Tabulka *document_category* obsahuje pouze *unikátní identifikátor* (primární klíč), *název kategorie* a *pozici* pro řazení dokumentů na stránkách. Tabulka *document* opět obsahuje *unikátní identifikátor* (primární klíč) a *unikátní identifikátor* tabulky *document_category*, tvořící cizí klíč. Do atributu *title* se ukládá zvolený název souboru, který se zobrazuje na stránkách. Do atributu *file_name* se zapíše název nahrávaného souboru. Do *file* pak unikátní název souboru, který se generuje tak, aby na serveru nebyly dva soubory se stejným názvem. Sloupec *type* uchovává typy internetového média (MIME). Na konci jsou atributy *position* pro pozicování, *date_add* a *date_upd* s datem a časem kdy byl soubor nahrán nebo upraven.

V systému se rozlišují dva *modely*. *Model* pro kategorie *DocumentCategory.php* a *model* pro samotné dokumenty *Document.php* s následujícím umístěním.

```
\data\app\bin\Document
```

Jednou z důležitých funkcí uvnitř *DocumentCategory.php* je funkce, která vrací seznam kategorií dokumentů.

```
public static function getList($limit = false){
    $data = DB::getInstance()-
>SelectMulti("document_category",false,["position","ASC"], $limit);
    $ret = [];
    if($data and is_array($data)){
        foreach($data as $v){
            $ret[] = new
DocumentCategory($v['id_document_category'],
          $v['name'], $v['position']);
```

```
        }
    }
    return $ret;
}
```

Funkce se pokusí načíst data z databáze a pokud uspěje, vytvoří instanci objektu a vrátí seznam kategorií.

Na vypsání samotných dokumentů z kategorie je určena funkce, která se nachází v *modelu Document.php*.

```
public static function getListByCategory($id_category) {
    $data = DB::getInstance()->SelectMulti("document", [
        "id_category" => $id_category
    ], ["position", "ASC"]);
    $ret = [];
    if ($data and is_array($data)) {
        foreach ($data as $v) {
            $ret[] = new Document($v['id_document'],
$v['id_category'], $v['title'], $v['file_name'], $v['file'],
$v['type'], $v['position'], $v['date_add'], $v['date_upd']);
        }
    }
    return $ret;
}
```

Princip je stejný jako u výpisu kategorií. Podle identifikátoru kategorie se vytvoří seznam odpovídajících výsledků, který se následně přes instanci objektu vypíše. Stažení souboru je realizováno funkcí *download* viz Příloha XI. Ta zkontroluje existenci souboru, následně nastaví hlavičku dokumentu a odešle soubor na výstup. Poté provede ukončení na straně serveru.

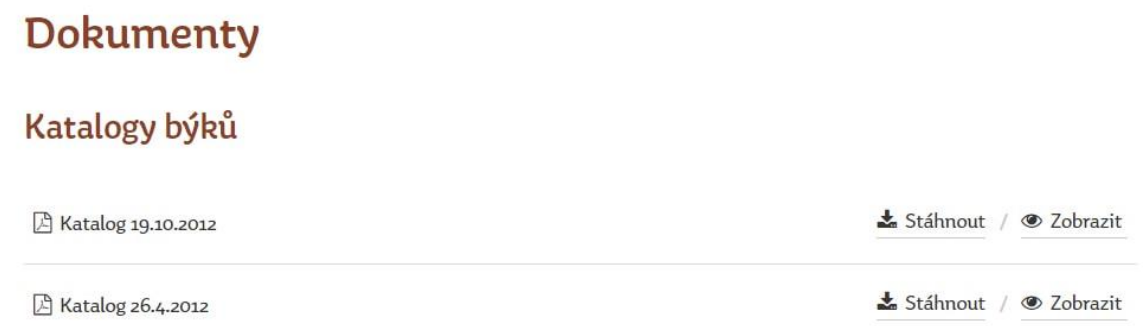
Administrace (viz Obr. 5.17) obsahuje všechny z naprogramovaných funkcí. Je zde tlačítko pro přidání, vyvolávající modalové okno. Lze také přepínat záložky kategorií a mazat či editovat jejich záznamy.

Obr. 5.17 Administrační rozhraní dokumentů

Na frontendu jsou dokumenty (viz Obr. 5.18) na samostatné stránce, kde mají návštěvníci možnost si je zobrazit v novém okně nebo stáhnout. Název staženého souboru odpovídá hodnotě nastavené v atributu *title*. Ikonky před názvem označují typ souboru. Očekávané jsou soubory ve formátu *.pdf*, *.png*, *.jpg* a *.jpeg*. U obrázků se zobrazí ikonka obrázku, u *.pdf* ikonka dokumentu PDF.

Takto vypadá funkce vracející hodnotu podle koncovky nahraného souboru, která tuto hodnotu vypíše do HTML kódu jako CSS třídu (této třídě byla v CSS přiřazena vhodná ikonka).

```
<td><i class="fa fa-{$doc->getFileExtension()}"></i> {$doc->getTitle()}</td>
```



Obr. 5.18 Frontendová podoba dokumentů

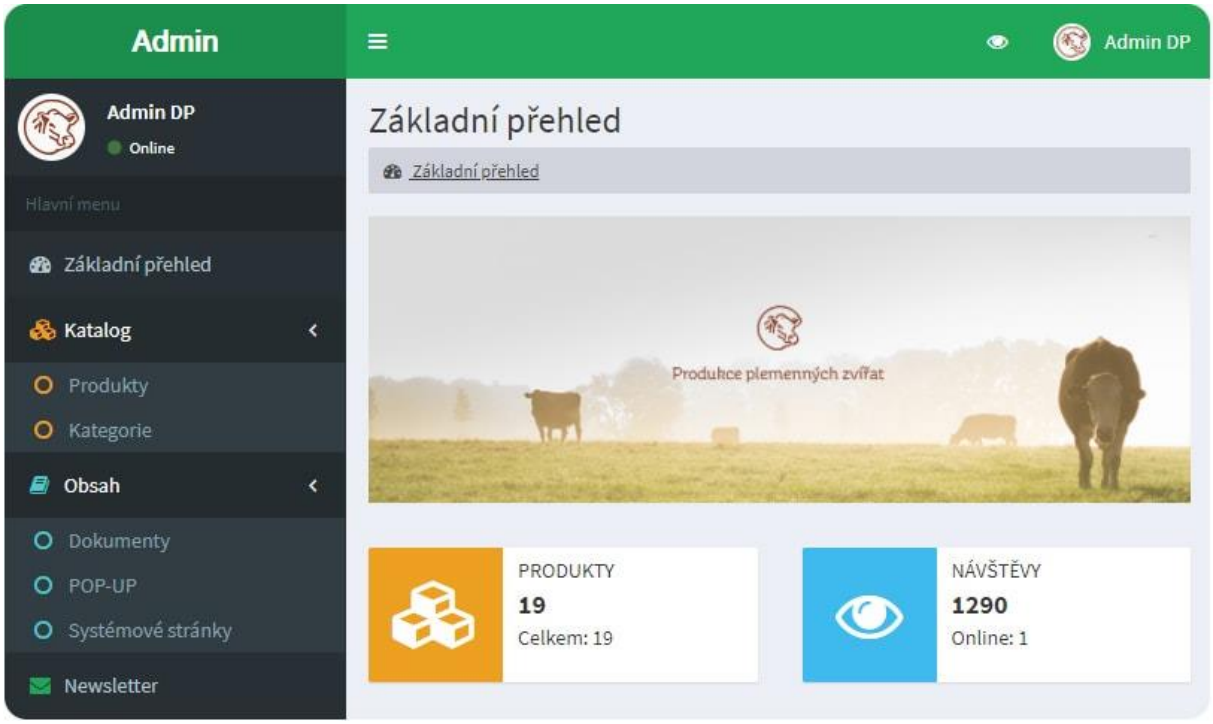
V šabloně byly využity i speciální vlastnosti *@first* a *@last* u cyklu *foreach*. Společně s prvním vypsaným záznamem se vypíše i nadpis a začátek tabulky. Následuje tělo cyklu, to vypíše řádky s dokumenty a na konci pomocí *@last* se tabulka ukončí.

```
{foreach Document\DocumentCategory::getList() as $cat}
     {foreach $cat->getFiles() as $doc}
          {if $doc@first}
               <h2>{$cat->getName()}</h2>
               <table class="table">
          {/if}
          <!-- tělo cyklu -->
          {if $doc@last}
               </table>
          {/if}
     {/foreach}
{/foreach}
```

5.6 Souhrn vlastní realizace

Výsledkem praktické části jsou webové stránky s administrací, na kterých byly realizovány všechny zadané náležitosti. Chovateli byl vytvořen přístup do administrace viz Obr. 5.19. Má zde možnost vkládat nové produkty, spravovat kategorie, dokumenty a newsletter, editovat texty systémových stránek, aktivovat a upravovat vyskakovací okno.

Dále zde má na úvodní stránce k dispozici přehled o celkovém počtu produktů, návštěv, historii svých přihlášení a poslední registrované návštěvníky v newsletteru.



Obr. 5.19 Úvodní stránka administračního rozhraní

5.7 Optimalizace

Kapitola je věnována základní optimalizaci pro vyhledávače, která obecně koresponduje s optimalizací stránek. Faktor, kterým je například rychlost načtení je současně jedním z faktorů pro kvalitní SEO optimalizaci.

Mezi další faktory patří kvalitní realizace responzivity, správná struktura úrovní nadpisů, vyplněné alternativní texty u obrázků, meta značky, informace pro vyhledávací roboty nebo soubor se strukturou stránek.

Tyto základní faktory byly při realizaci splněny. Pro ilustraci si prezentujeme výsledky časů potřebných pro načtení stránky v různých situacích.

První případ uvádí situaci, kdy návštěvník přistupuje na stránky v desktopové verzi, přičemž načítané obrázky jsou komprimované, skripty a CSS též. Výsledný čas viz Příloha IX se pohybuje kolem jedné sekundy.

Druhá situace simuluje uživatele, který na stránky přistupuje z mobilního telefonu viz Obr. 5.20. V takovém případě nemusí být připojen k WiFi ani 4G LTE. Na lokálním serveru byla pomocí browsersync prostředí nastavena maximální rychlost připojení na 750 kb/s. Výsledkem jsou časy pohybující se kolem tří sekund viz Příloha X.

Obr. 5.20 Zobrazení na telefonu

6 Závěr

Veškeré náležitosti zadané chovatelem byly splněny, zároveň byly realizovány i autorem doporučené funkce, které se nevyskytovaly v původním zadání. Mezi náležitosti patřila realizace grafického návrhu a následná realizace stránek za použití vybraných technologií. Byl vytvořen modulární systém na principu internetového obchodu s možností pracovat s kategoriemi a produkty. Dále modul určený pro sbírání e-mailů a telefonních čísel od zákazníku. Aktivovatelné vyskakovací okno s informacemi o otevírací době chovatelova řeznictví a modul pro správu dokumentů. Výsledné řešení bylo podrobeno optimalizačním testům zaměřeným zejména na rychlost načítání stránek.

Vlastním přínosem je tak vytvoření vhodného nástroje, pomocí kterého nejenže může chovatel nabízet své produkty, ale také je katalogizovat. Tím získá dostatečný přehled o aktuálním stavu vlastněných jedinců včetně jejich fotodokumentace i historie již prodaných kusů.

Dále může v rámci jednoho systému centralizovat dokumenty, jako jsou výběrové protokoly a katalogy býků. Jeho stávající i noví zákazníci se mohou do tohoto systému zaregistrovat. V budoucnu je tak očekáván zvýšený odbyt chovatele a lepší prodejní výsledky jeho řeznictví.

7 Závěrečná doporučení

Z pohledu zlepšování nabízených služeb by samotná realizace neměla být pro chovatele konečnou fází. Systémové řešení je koncipováno tak, aby bylo jeho rozšiřování o další funkce zcela bezproblémové. Mimo zadání byla připravena sada řešení, pomocí kterých lze stávající systém rozvíjet do podoby plnohodnotného internetového obchodu.

Nabízí se také možnost zvýšit zákaznickou základnu díky systémové podpoře více jazyků a měn. Nemělo by být upuštěno od plánovaného zavedení produktových štítků, které budou nést informace o stáří, případně váze jedince.

Pro zvýšení prodeje je doporučena nejen postupná optimalizace stránek, ale také sledovaní aktivit návštěvníků pro postupné vylepšování konverzních cest.

Terminologický slovník

Termín	Zkratka	Význam
Asynchronous JavaScript and XML	AJAX	AJAX umožňuje, aby stránka kontaktovala server a obdržela od něj libovolná data v XML, bez toho, aby se musela celá znovu nahrávat
Application Programming Interface	API	rozhraní pro programování funkcí
backend	BE	administrační část webových stránek
call-to-action	CTA	prvek s výzvou k akci návštěvníka stránek
Common Language Runtime	CLR	prostředí, které řídí provádění kódu a poskytuje služby pro vývoj aplikací [54]
Content Management System	CMS	systém určený ke správě obsahu (webových stránek)
cookies		krátké textové soubory vytvářené webovým serverem a ukládané v počítači prostřednictvím prohlížeče
Cascading Style Sheets	CSS	jazyk popisující způsob zobrazení elementů na stránkách.
debugger		nástroj používaný pro hledání chyb při programování
framework		aplikační rámec, který lze využívat k programování/kódování.
frontend	FE	část webových stránek, která se zobrazuje návštěvníkovi
GNU General Public License	GNU GPL	licence, pod kterou lze programy volně používat, modifikovat a šířit, pokud to bude bezplatně a kód zůstane otevřený. Jména autorů se nesmí odstraňovat.
grid		základní mřížka webových stránek ulehčující kodérské práce
Graphical User Interface	GUI	grafické uživatelské rozhraní
hexadecimal	HEX	šestnáctková soustava, ve které lze zapisovat barvy

HyperText Markup Language	HTML	značkovací jazyk pro tvorbu webových stránek
JavaScript Object Notation	JSON	datový formát a způsob zápisu do polí nebo objektů
layout		základní rozvržení prvků na webové stránce nebo soubor obsahující určitou strukturu
Long Term Evolution	LTE	technologie vysokorychlostního Internetu
Multipurpose Internet Mail Extensions	MIME	standardizovaná cesta k určení povahy a formátu dokumentu
MIT Licence		licence pod, kterou lze programy volně používat a šířit (i jako součást proprietárního software), povinností je přiložení kopie licence a uvedení jména autora
mockup		náhled v reálném použití
Model-View-Controller	MVC	architektura, ve které je oddělen datový model, uživatelské rozhraní a řídící logika
MySQL		multiplatformní relační databázový systém
newsletter		informační zpravodaj, do kterého se mohou uživatelé přihlásit
Node Package Manager	NPM	správce balíčků pro Node.js
Open Source Software	OSS	software, který mí otevřený zdrojový kód
Hypertext Preprocessor	PHP	skriptovací programovací jazyk
pixel	px	jednotka přestavující jeden bod na monitoru
plugin		část softwaru (modul), který nepracuje samostatně
prefix		část, která se vkládá před slovo (předpona)
retina		název pro displej s vysokou hustotou pixelů

Search Engine Optimization	SEO	optimalizace webových stránek pro vyhledávače
User Interface	UI	uživatelské rozhraní
Uniform Resource Locator	URL	slouží k jednoznačné specifikaci umístění souborů (webových stránek) na Internetu
vektor		matematicky popsatelná veličina, vektorovému grafickému výstupu lze měnit velikost bez ztráty kvality
World Wide Web Consortium	W3C	mezinárodní sdružení dohlížející na webové standardy
webhosting		služba nabízející pronájem prostoru pro webové stránky
Wireless Fidelity	WiFi	technologie pro bezdrátový přenos
Cross-site scripting	XSS	metoda využívající bezpečnostní chyby webových stránek

Seznam použitých zdrojů

1. Smola, Martin. HTML5: co přináší a proč se o něj zajímat. *root.cz.* [Online] Internet Info, s.r.o, 29. 8 2012. [Citace: 6. 1 2018.] Dostupné z: www.root.cz/clanky/html5-co-prinasi-a-proc-se-o-nej-zajimat.

2. Sládek, Jan. Webdesignérův průvodce po HTML5 – nová sémantika. *zdoják.cz.* [Online] Devel.cz Lab s.r.o., 6. 1 2010. [Citace: 6. 1 2018.] Dostupné z: www.zdrojak.cz/clanky/webdesigneruv-pruvodce-po-html5-nova-semantika.

3. Elizabeth Castro, Bruce Hyslop. *HTML5 a CSS3 - Názorný průvodce tvorbou WWW stránek.* místo neznámé : Computer Press, 2012. ISBN 978-80-251-3733-8.

4. Prokop, Marek. Přehled standardů W3C. *interval.cz.* [Online] 27. 5 2002. [Citace: 6. 1 2018.] Dostupné z: www.interval.cz/clanky/prehled-standardu-w3c.

5. Jahoda, Bohumil. HTML značky a . *Je čas.* [Online] 3. 2 2016. [Citace: 6. 1 2018.] Dostupné z: www.jecas.cz/div-span .

6. Čápka, David. Struktura HTML dokumentu - Český HTML 5 manuál. *ITnetwork.cz.* [Online] 2013. [Citace: 7. 1 2018.] Dostupné z: www.itnetwork.cz/html-css/html-manual/struktura/html-struktura-stranky-cesky-manual.

7. Pilgrim, Mark. *DETEKUJEME PODPORU HTML5.* [překl.] Martin Hassman. 2010. 978-0596806026.

8. *HTML Living Standard.* [Online] [Citace: 7. 1 2018.] Dostupné z: www.html.spec.whatwg.org/multipage/media.html#video.

9. Canvas – říkejme tomu plocha na kreslení. *zdroják.cz.* [Online] Devel.cz Lab s.r.o., 23. 3 2012. [Citace: 8. 1 2018.] Dostupné z: www.zdrojak.cz/clanky/canvas-rikejme-tomu-plocha-na-kresleni.

10. Je čas. *HTML <canvas>.* [Online] 4. 1 2016. [Citace: 8. 1 2018.] Dostupné z: www.jecas.cz/canvas.

11. Kučera, Vlastislav. *MATEMATIKA–FYZIKA–INFORMATIKA.* [Online] 2014. [Citace: 14. 1 2018.] Dostupné z: www.mfi.upol.cz/files/23/2302/mfi_2302_140_147.pdf.

12. Nothrem. FLEXBOX: LAYOUT BUDOUCNOSTI. *CSS3 Tipy a Triky.* [Online] 11. 24 2014. [Citace: 19. 1 2018.] Dostupné z: www.css.chobits.ch/flexbox.

13. Gasston, Peter. *CSS3 - Průvodce vývojáře moderního webdesignu.* místo neznámé : CPRESS, 2016. ISBN: 978-80-251-4641-5.

14. CSS Vendor Prefixes. *bitsofcode.* [Online] 12. 5 2015. [Citace: 17. 1 2018.] Dostupné z: www.bitsofco.de/css-vendor-prefixes.

15. Sellier, Alexis. {less}. *Dynamický jazyk pro tvorbu stylesheetů.* [Online] [Citace: 18. 1 2018.] Dostupné z: www.lesscss.cz.

16. Michálek, Martin. Průvodce CSS preprocesory: který vybrat? *Vzhůru dolů.* [Online] 1. 4 2014. [Citace: 20. 1 2018.] Dostupné z: www.vzhurudolu.cz/blog/15-css-preprocesory-4.

17. It's CSS, with just a little more. *{less}.* [Online] [Citace: 22. 1 2018.] Dostupné z: www.lesscss.org/functions.

18. Coyier, Chris. CSS-TRICKS. *Reference Imports in LESS (are kinda cool).* [Online] 25. 9 2015. [Citace: 22. 1 2018.] Dostupné z: www.css-tricks.com/reference-imports-in-less-are-kinda-cool.

19. Giraudel, Hugo. Umíněný průvodce pro psaní rozumného, udržovatelného a škálovatelného Sass. *Sass Guidelines.* [Online] [Citace: 23. 1 2018.] https://sass-guidelin.es/cz/.

20. The !default and !global flags. *anotheruiguy.* [Online] [Citace: 23. 1 2018.] Dostupné z: www.anotheruiguy.gitbooks.io/sassintherealworld_book-i/handy-tools/default-flag.html.

21. Martsoukos, George. Understanding Variable Scope in Sass. *envatotuts+.* [Online] 7. 5 2015. [Citace: 23. 1 2018.] Dostupné z: www.webdesign.tutsplus.com/articles/understanding-variable-scope-in-sass--cms-23498.

22. Coyier, Chris. Sass vs. LESS. *CSS-Tricks.* [Online] 16. 5 2012. [Citace: 23. 1 2018.] Dostupné z: www.css-tricks.com/sass-vs-less/.

23. stylus/nib. *github.* [Online] 25. 6 2014. [Citace: 26. 1 2018.] Dostupné z: www.github.com/stylus/nib/blob/master/lib/nib/size.styl.

24. Michálek, Martin. Bootstrap a front-end frameworky. *Webstory.* [Online] 21. 1 2014. [Citace: 28. 1 2018.] Dostupné z: www.webstory.cz/bootstrap-a-front-end-frameworky.

25. Teršel, Jíří. CSS frameworky. *Masarykova univerzita.* [Online] 2010. [Citace: 29. 1 2018.] Dostupné z: www.fi.muni.cz/~xobsivac/PV219/prezentace10/css.frameworky.pdf.

26. twbs/bootstrap. *github.* [Online] [Citace: 28. 1 2018.] Dostupné z: www.github.com/twbs/bootstrap.

27. zurb/foundation-sites. *github.* [Online] [Citace: 28. 1 2018.] Dostupné z: www.github.com/zurb/foundation-sites.

28. Bootstrap 3 vs. Foundation 5: Which Front-end Framework Should You Use? *codementor.* [Online] 23. 6 2015. [Citace: 29. 1 2018.] Dostupné z: www.codementor.io/codementorteam/bootstrap-3-vs-foundation-5-which-front-end-framework-should-you-use-8s90mhsgn .

29. Global Styles. *ZURB foundation.* [Online] [Citace: 2. 2 2018.] Dostupné z: www.foundation.zurb.com/sites/docs/global.html.

30. PHP (1) - Historie a budoucnost. *Linuxsoft.* [Online] 27. 5 2004. [Citace: 4. 2 2018.] Dostupné z: www.linuxsoft.cz/article.php?id_article=171 .

31. Ponkrác, Miroslav. *PHP a MySQL bez předchozích znalostí.* místo neznámé : Computer Press, 2007. 978-80-251-1758-3.

32. Čápka, David. 1. díl - Úvod do ASP.NET. *ITnetwork.* [Online] [Citace: 4. 2 2018.] Dostupné z: www.itnetwork.cz/csharp/asp-net/tutorial-uvod-do-asp-dot-net.

33. Cross platform, open source .NET framework. *Mono.* [Online] [Citace: 4. 2 2018.] Dostupné z: www.mono-project.com.

34. Haramule, Václav. .NET Core – Úvod do platformy. *SKELETON software.* [Online] 20. 3 2017. [Citace: 4. 2 2018.] Dostupné z: www.skeleton.cz/net-core-uvod-do-platformy.

35. WORDPRESS – ČESKÁ PODPORA. *cwordpress.* [Online] [Citace: 6. 2 2018.] Dostupné z: www.cwordpress.cz.

36. Co umí WordPress. *coumiwp.* [Online] [Citace: 7. 2 2018.] Dostupné z: www.coumiwp.cz/co-je-wordpress.

37. Codex. *WordPress.org.* [Online] [Citace: 7. 2 2018.] Dostupné z: www.codex.wordpress.org/Database_Description.

38. Herout, Tomáš. Co je to Joomla, k čemu slouží a jak používá číslování verzí. *helpmark.* [Online] 20. 4 2017. www.helpmark.cz/navody/navody-joomla/uvod-joomla/324-co-je-to-joomla.

39. Technical requirements. *Joomla! Documentation™.* [Online] [Citace: 9. 2 2018.] Dostupné z: www.docs.joomla.org/Technical_requirements.

40. Total extensions. *Joomla! Extensions Directory™.* [Online] [Citace: 9. 2 2018.] Dostupné z: www.extensions.joomla.org.

41. Bezpečnost Drupalu. *Drupal arts.* [Online] [Citace: 11. 2 2018.] Dostupné z: www.drupalarts.cz/bezpecnost-drupalu.

42. O Drupalu. *Drupal.* [Online] [Citace: 14. 2 2018.] Dostupné z: www.drupal.cz/o-drupalu.

43. Výběr vhodného šablonovacího systému pro webintegrační projekt na platformě PHP. *Webová integrace.* [Online] 5. 5 2014. [Citace: 17. 2 2018.] Dostupné z: www.web-integration.info/cs/blog/vyber-vhodneho-sablonovaciho-systemu-pro-webintegracni-projekt-na-platforme-php.

44. MVC aplikace & presentery. *Nette Documentation.* [Online] [Citace: 20. 2 2018.] Dostupné z: www.doc.nette.org/cs/2.4/presenters#toc-model-view-controller-mvc.

45. 1. díl - Úvod do Nette frameworku pro PHP. *ITnetwork.* [Online] [Citace: 19. 2 2018.] Dostupné z: www.itnetwork.cz/php/nette/zaklady/uvod-do-php-frameworku-nette.

46. nette/sandbox. *github.* [Online] [Citace: 20. 2 2018.] https://github.com/nette/sandbox.

47. Nothrem. LATTE JAKO OD BARISTKY. *CSS3 Tipy a Triky.* [Online] 7. 6 2017. [Citace: 20. 2 2018.] Dostupné z: www.css.chobits.ch/latte.

48. Zend\View Quick Start. *Zend Framework.* [Online] [Citace: 21. 2 2018.] Dostupné z: www.framework.zend.com/manual/2.4/en/modules/zend.view.quick-start.html.

49. Smarty Template Engine 1.díl - úvod. *Banan webhosting.* [Online] [Citace: 21. 2 2018.] Dostupné z: www.banan.cz/serialy/PHP-a-HTML/Smarty-Template-Engine-1-dil-uvod .

50. Hayder Hasin, J. P. Maia, Gheorghe Lucian. *Smarty PHP Template Programming And Applications.* místo neznámé : Packt Publishing, 2006. 978-1904811404.

51. smarty template engine. *smarty.* [Online] [Citace: 25. 2 2018.] Dostupné z: www.smarty.net/docsv2/en/what.is.smarty.tpl.

52. Bernard, Borek. Úvod do architektury MVC. *zdrojak.* [Online] 7. 5 2009. [Citace: 26. 2 2018.] Dostupné z: www.zdrojak.cz/clanky/uvod-do-architektury-mvc.

53. documentation. *smarty template engine.* [Online] [Citace: 2. 3 2018.] Dostupné z: www.smarty.net/docsv2/en/language.function.if.tpl .

54. Steven Roman, Paul Lomax, Ron Petrusha. *VB.NET Language in a Nutshell, Second Edition.* 2002. 0596003080.

55. Pilgrim, Mark. *DETEKUJEME PODPORU HTML5.* [překl.] Martin Hassman. 2010. 978-0596806026.

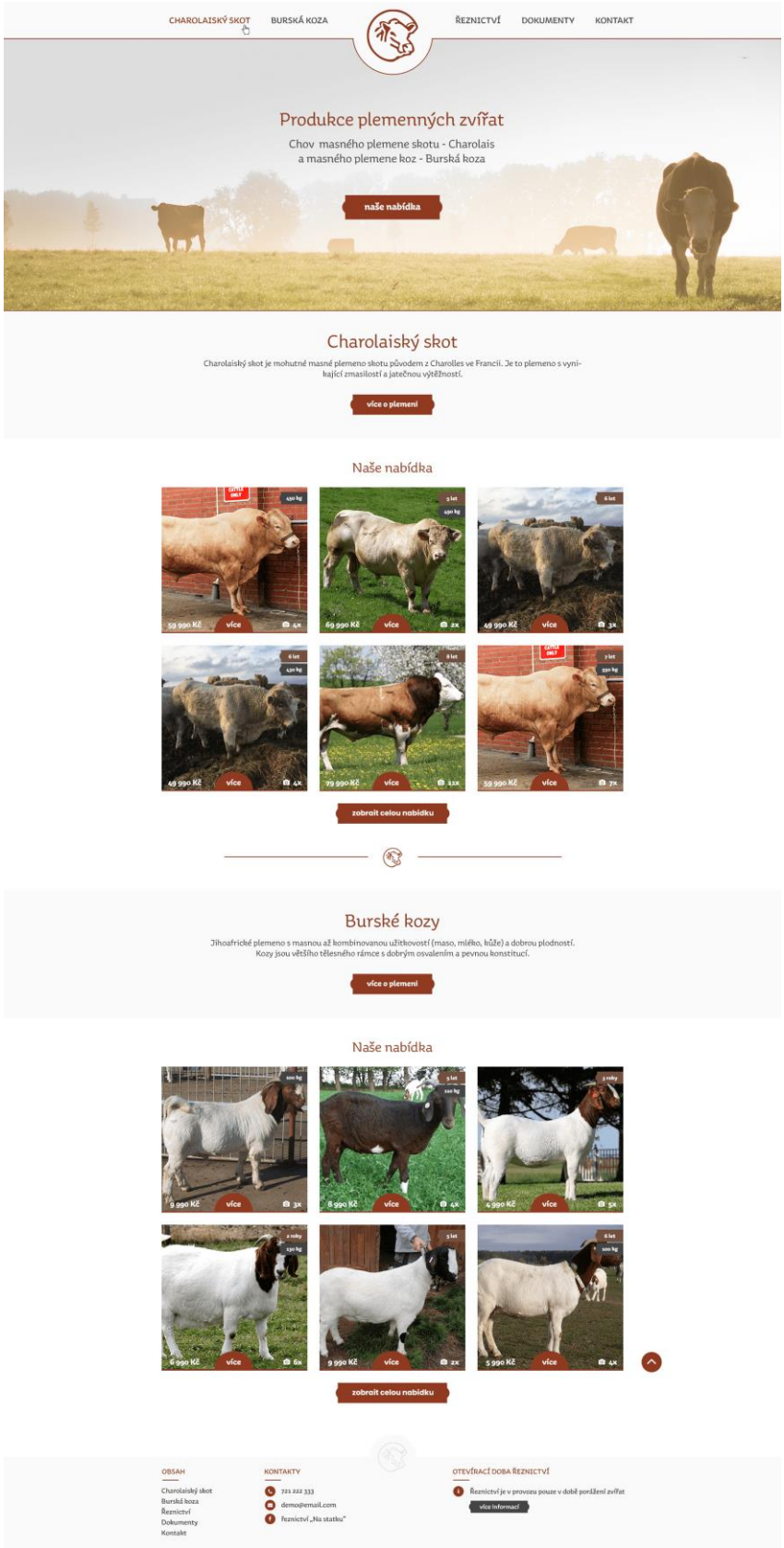
56. Dalbert, Jerome. stack overflow. *Controller belongs to the Presentation layer?* [Online] 16. 9 2012. [Citace: 16. 2 2018.] https://stackoverflow.com/questions/12429729/controller-belongs-to-the-presentation-layer.

Seznam obrázků

Seznam tabulek

Seznam příloh

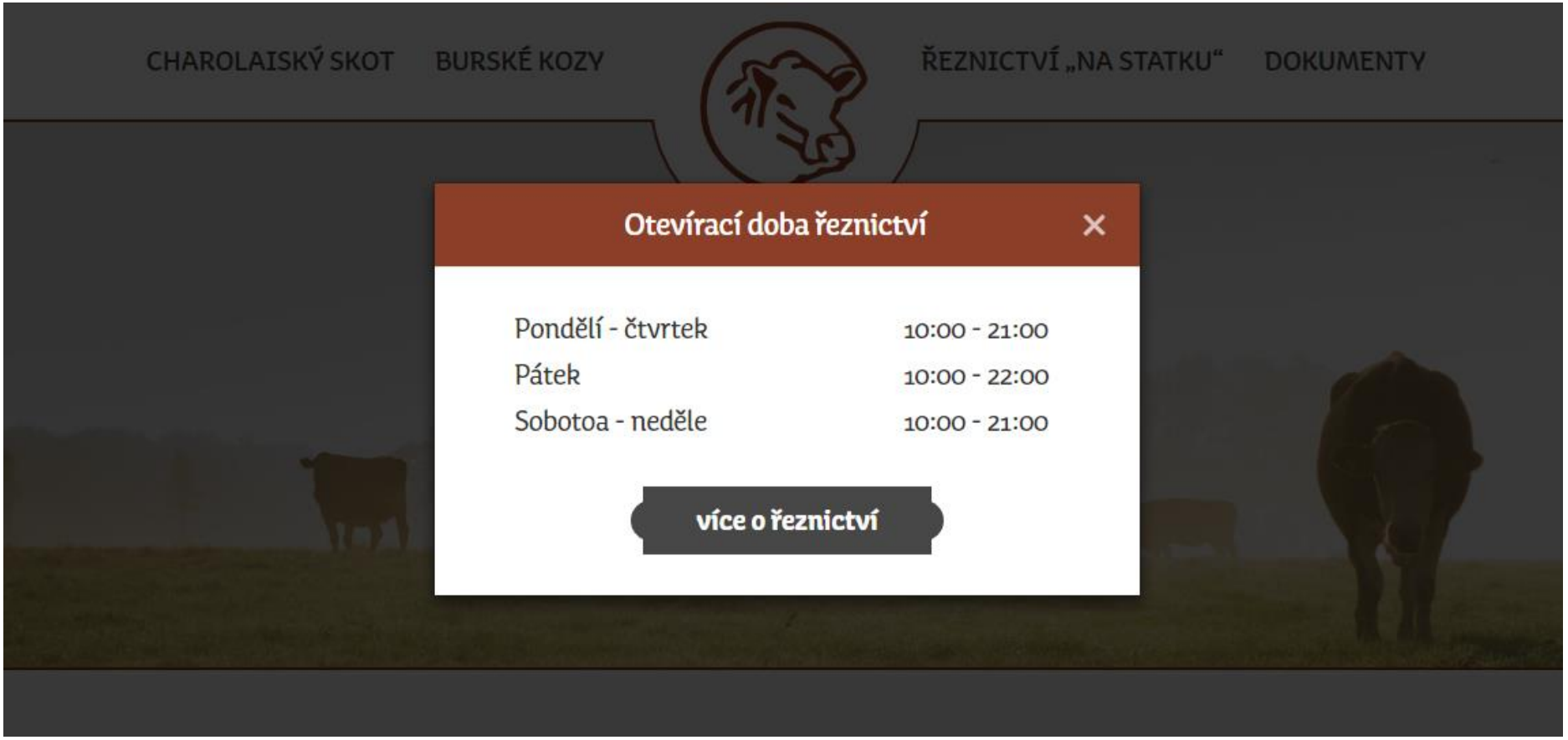
Příloha I


CHAROLAISKÝ SKOT
BURSKÁ KOZA
ŘEZNICTVÍ
DOKUMENTY
KONTAKT
Produkce plemenných zvířat
Chov masného plemene skotu - Charolais
a masného plemene koz - Burská koza
naše nabídka
Charolaiský skot
Charolaiský skot je mohutné masné plemeno skotu původem z Charolles ve Francii. Je to plemeno s vyni-
kající zmasilostí a jatečnou výtěžností.
více o plemeni
Naše nabídka
zobrait celou nabídku
Burské kozy
Jihoafrické plemeno s masnou až kombinovanou užitkovostí (maso, mléko, kůže) a dobrou plodností.
Kozy jsou většího tělesného rámce s dobrým osvalením a pevnou konstitucí.
více o plemeni
Naše nabídka
zobrait celou nabídku
OBSAH
Charolaiský skot
Burská koza
Řeznictví
Dokumenty
Kontakt
KONTAKTY
721 222 333
demo@email.com
Řeznictví „Na statku"
OTEVÍRACÍ DOBA ŘEZNICTVÍ
Řeznictví je v provozu pouze v době porážení zvířat
více informací

Příloha II


mockup
iMac
CHAROLAISKÝ SKOT
BURSKÁ KOZA
ŘEZNICTVÍ
DOKUMENTY
KONTAKT
Produkce plemenných zvířat
Chov masného plemene skotu - Charolais
a masného plemene koz - Burská koza
naše nabídka
Charolaiský skot
více o plemeni
mockup
MacBook
Produkce plemenných zvířat
Charolaiský skot
Naše nabídka
MacBook Air

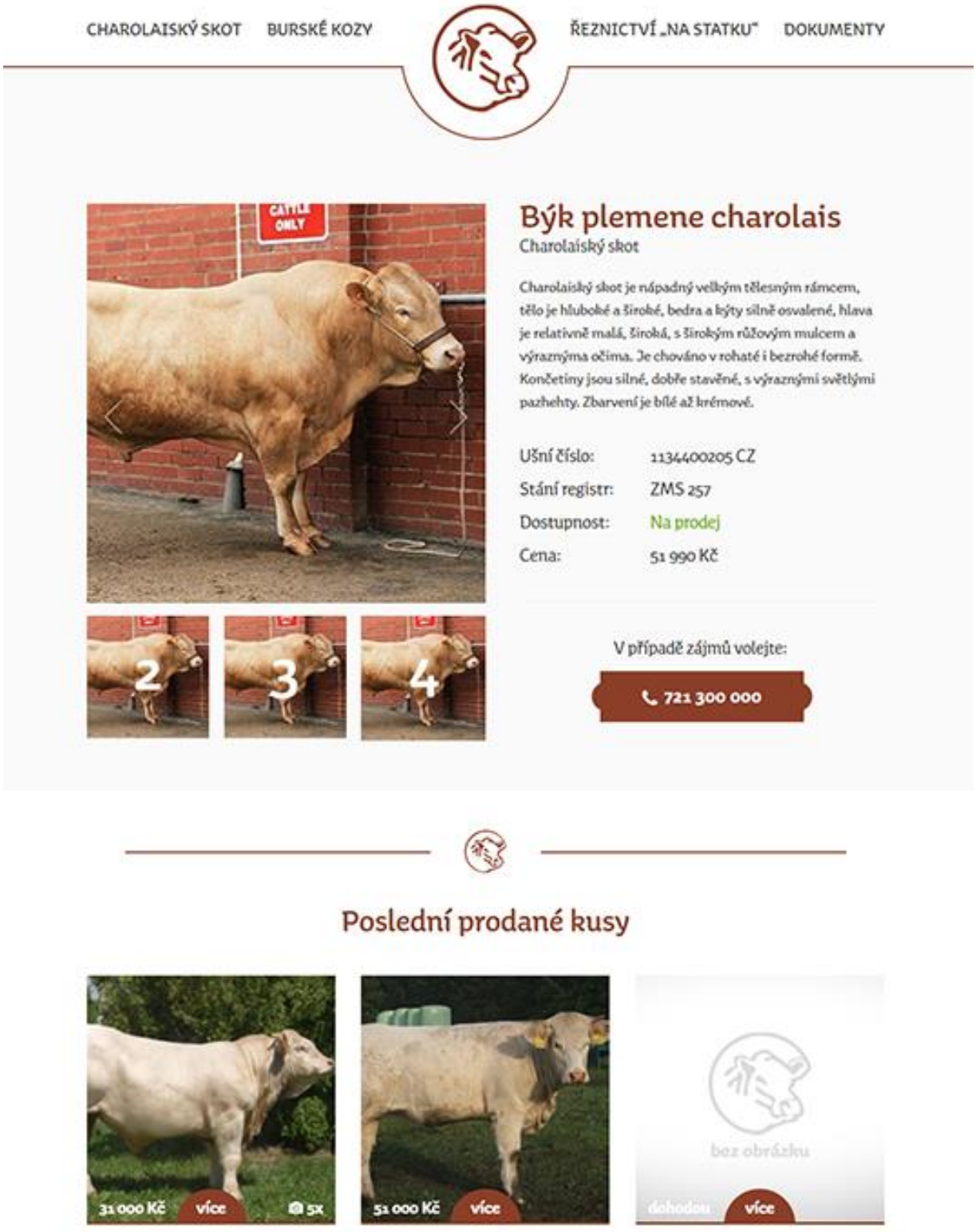
Příloha III


CHAROLAISKÝ SKOT
BURSKÉ KOZY
ŘEZNICTVÍ „NA STATKU“
DOKUMENTY
Otevírací doba řeznictví
Pondělí - čtvrtek 10:00 - 21:00
Pátek 10:00 - 22:00
Sobotoa - neděle 10:00 - 21:00
více o řeznictví

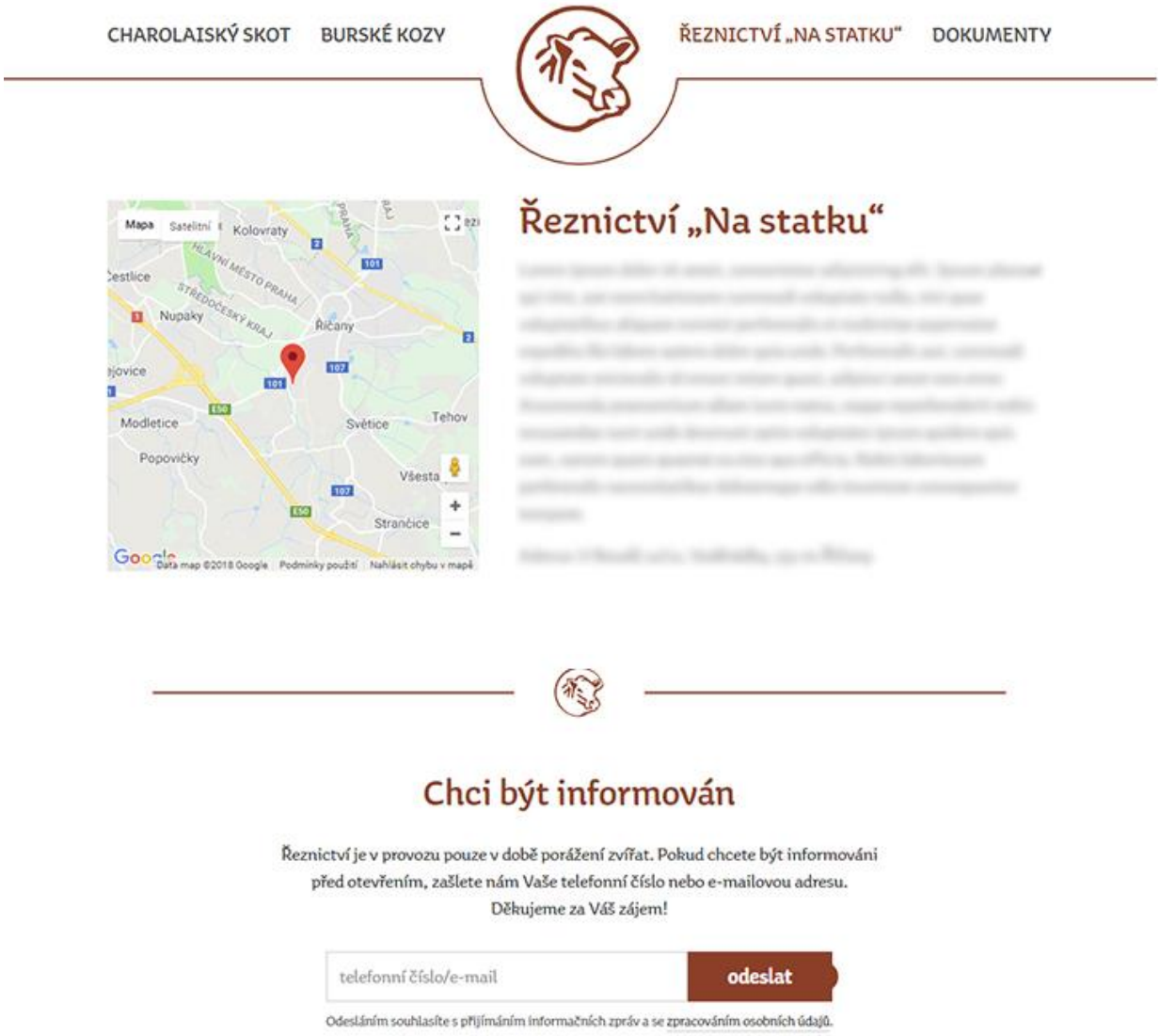
Příloha IV


CHAROLAISKÝ SKOT
BURSKÉ KOZY
ŘEZNICTVÍ „NA STATKU“
DOKUMENTY
Charolaiský skot
Charolaiský skot je mohutné masné plemeno skotu původem z Charolles ve Francii. Je to plemeno s vynikající zmasilostí a jatečnou výtěžností, jeho slabinou je naopak větší procento obtížných porodů. Je to jedno z nejčastěji chovaných masných plemen na světě, je nejpočetnějším masným plemenem ve Francii i v České Republice.
99 900 Kč
více
51 990 Kč
více
6x
39 900 Kč
více
51 000 Kč
více
7x
36 500 Kč
více
6x
29 900 Kč
více
4x

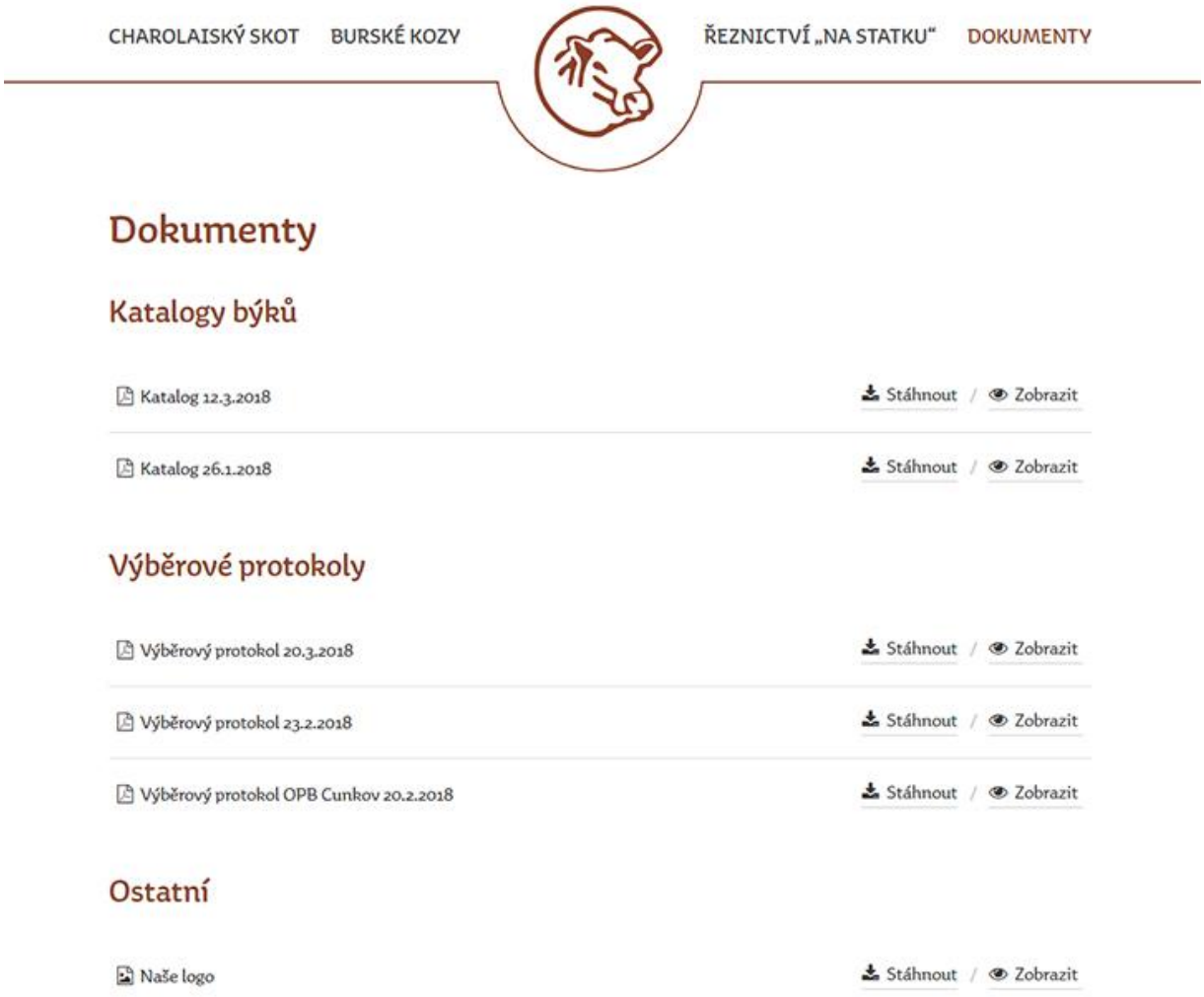
Příloha V


CHAROLAISKÝ SKOT
BURSKÉ KOZY
ŘEZNICTVÍ „NA STATKU"
DOKUMENTY
Býk plemene charolais
Charolaiský skot
Charolaiský skot je nápadný velkým tělesným rámcem, tělo je hluboké a široké, bedra a kýty silně osvalené, hlava je relativně malá, široká, s širokým růžovým mulcem a výraznýma očima. Je chováno v rohaté i bezrohé formě. Končetiny jsou silné, dobře stavěné, s výraznými světlými pazhehty. Zbarvení je bílé až krémové.
Ušní číslo: 1134400205 CZ
Stání registr: ZMS 257
Dostupnost: Na prodej
Cena: 51 990 Kč
V případě zájmů volejte:
721 300 000
Poslední prodané kusy
31 000 Kč
více
5x
51 000 Kč
více
bez obrázku
dohodou
více

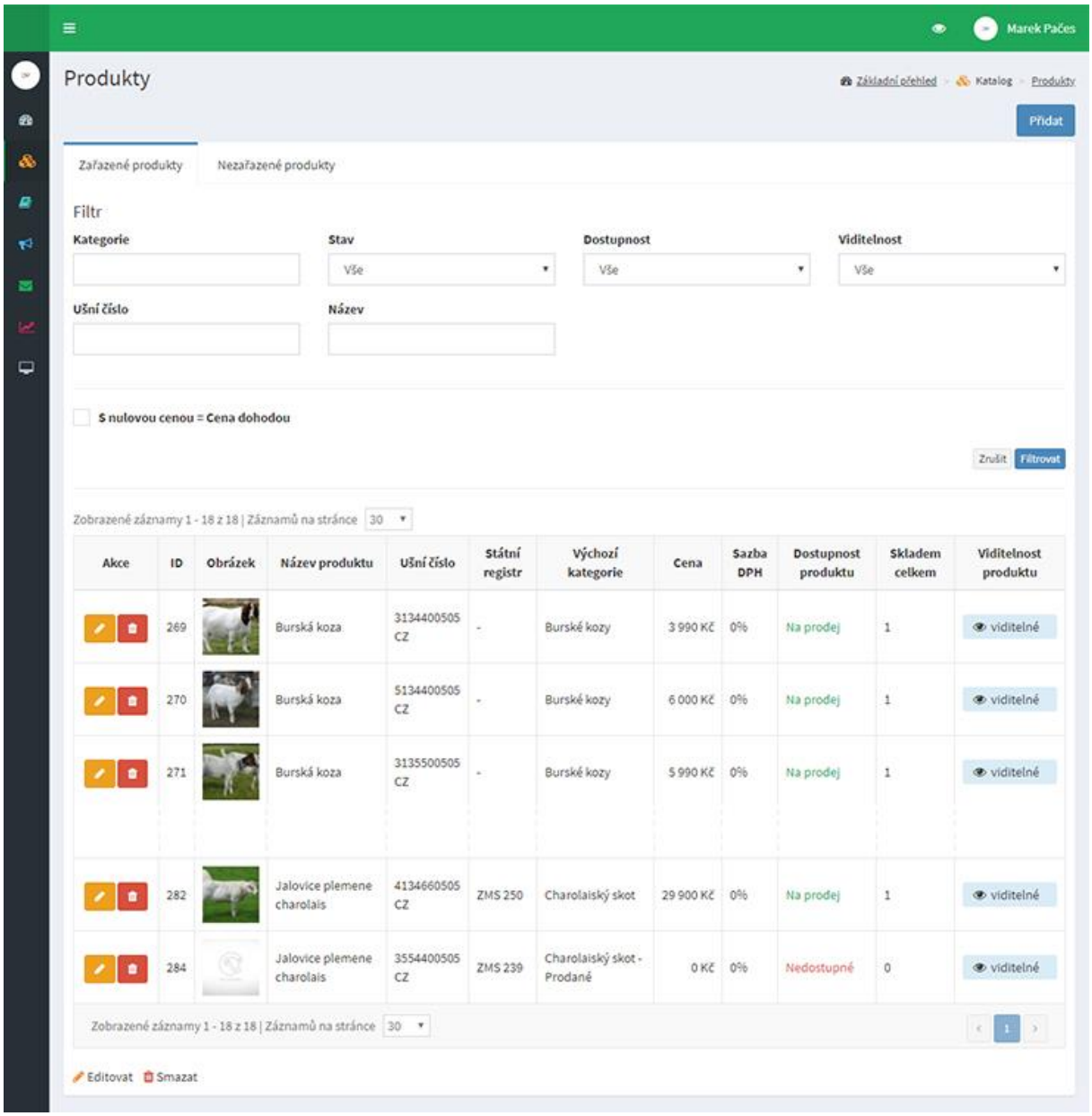
Příloha VI


CHAROLAISKÝ SKOT
BURSKÉ KOZY
ŘEZNICTVÍ „NA STATKU“
DOKUMENTY
Mapa
Satelitní
Kolovraty
Nupaky
Říčany
Modletice
Světice
Tehov
Popovičky
Všesta
Strančice
Data map ©2018 Google
Podmínky použití
Nahlásit chybu v mapě
Řeznictví „Na statku“
Chci být informován
Řeznictví je v provozu pouze v době porážení zvířat. Pokud chcete být informováni před otevřením, zašlete nám Vaše telefonní číslo nebo e-mailovou adresu.
Děkujeme za Váš zájem!
telefonní číslo/e-mail
odeslat
Odesláním souhlasíte s přijímáním informačních zpráv a se zpracováním osobních údajů.

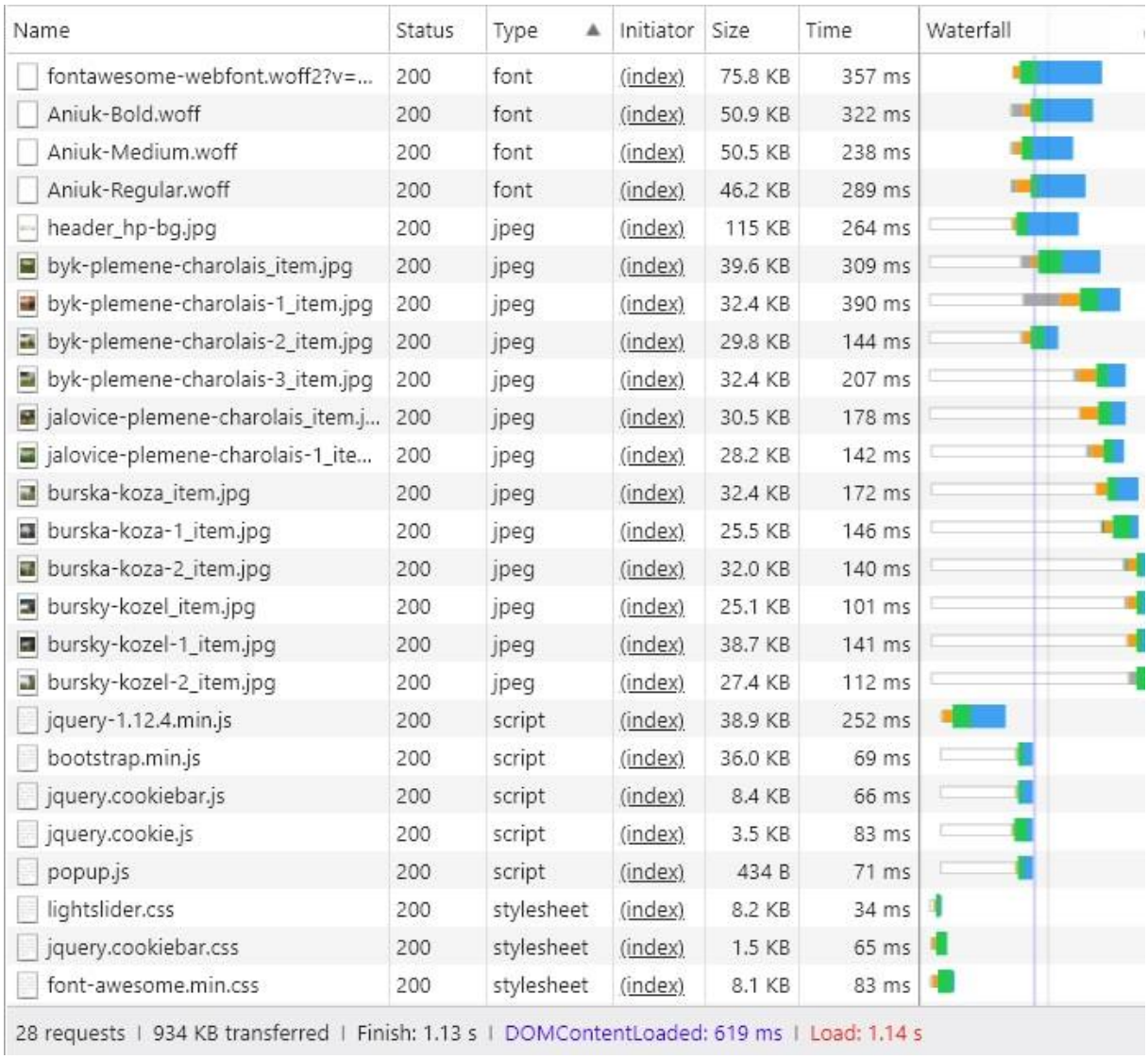
Příloha VII


CHAROLAISKÝ SKOT
BURSKÉ KOZY
ŘEZNICTVÍ „NA STATKU“
DOKUMENTY
Dokumenty
Katalogy býků
Katalog 12.3.2018
Stáhnout / Zobrazit
Katalog 26.1.2018
Stáhnout / Zobrazit
Výběrové protokoly
Výběrový protokol 20.3.2018
Stáhnout / Zobrazit
Výběrový protokol 23.2.2018
Stáhnout / Zobrazit
Výběrový protokol OPB Cunkov 20.2.2018
Stáhnout / Zobrazit
Ostatní
Naše logo
Stáhnout / Zobrazit

Příloha VIII



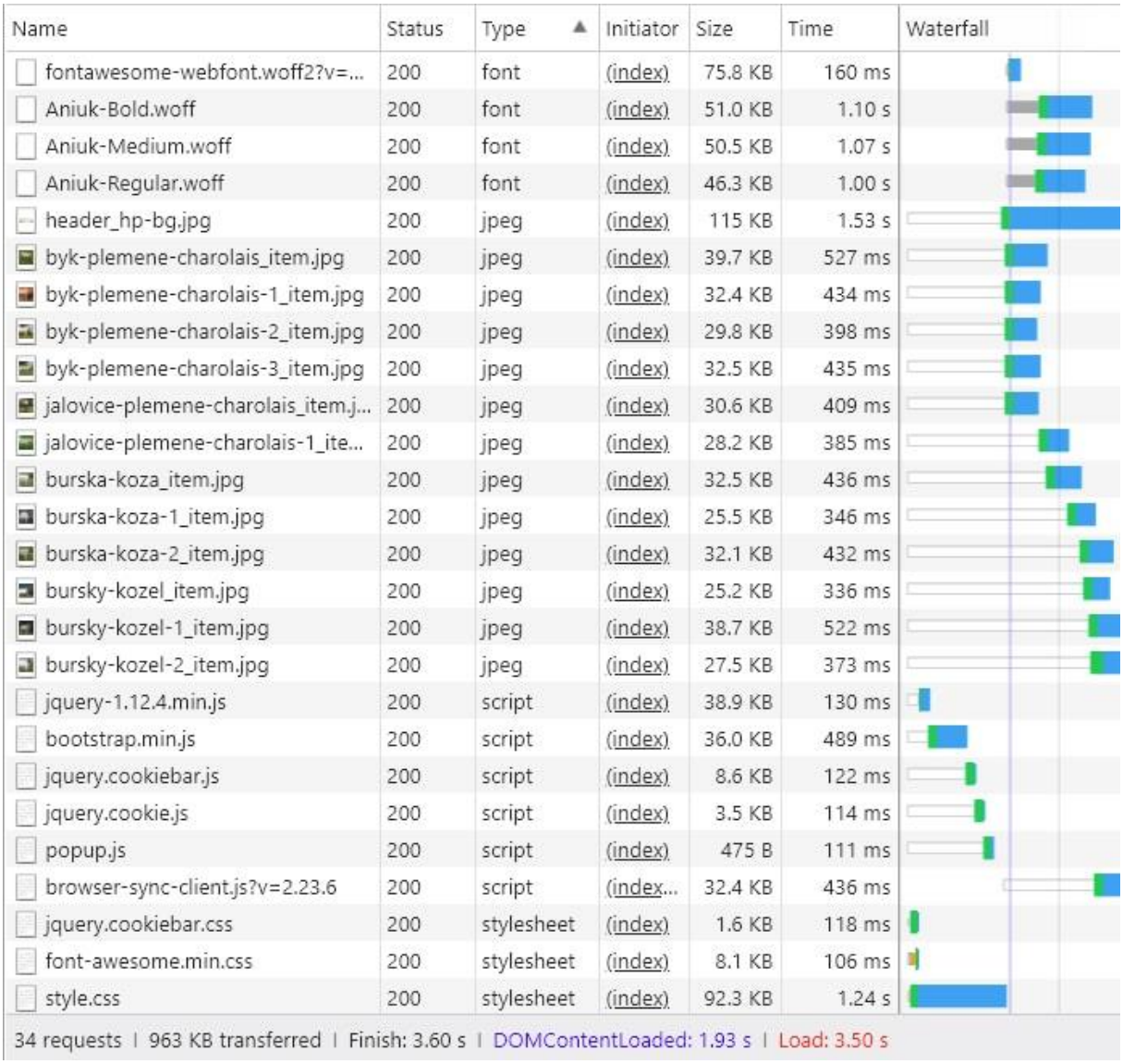
Příloha IX



Name	Status	Type	Initiator	Size	Time	Waterfall
fontawesome-webfont.woff2?v=...	200	font	(index)	75.8 KB	357 ms	
Aniuk-Bold.woff	200	font	(index)	50.9 KB	322 ms	
Aniuk-Medium.woff	200	font	(index)	50.5 KB	238 ms	
Aniuk-Regular.woff	200	font	(index)	46.2 KB	289 ms	
header_hp-bg.jpg	200	jpeg	(index)	115 KB	264 ms	
byk-plemene-charolais_item.jpg	200	jpeg	(index)	39.6 KB	309 ms	
byk-plemene-charolais-1_item.jpg	200	jpeg	(index)	32.4 KB	390 ms	
byk-plemene-charolais-2_item.jpg	200	jpeg	(index)	29.8 KB	144 ms	
byk-plemene-charolais-3_item.jpg	200	jpeg	(index)	32.4 KB	207 ms	
jalovice-plemene-charolais_item.j...	200	jpeg	(index)	30.5 KB	178 ms	
jalovice-plemene-charolais-1_ite...	200	jpeg	(index)	28.2 KB	142 ms	
burska-koza_item.jpg	200	jpeg	(index)	32.4 KB	172 ms	
burska-koza-1_item.jpg	200	jpeg	(index)	25.5 KB	146 ms	
burska-koza-2_item.jpg	200	jpeg	(index)	32.0 KB	140 ms	
bursky-kozel_item.jpg	200	jpeg	(index)	25.1 KB	101 ms	
bursky-kozel-1_item.jpg	200	jpeg	(index)	38.7 KB	141 ms	
bursky-kozel-2_item.jpg	200	jpeg	(index)	27.4 KB	112 ms	
jquery-1.12.4.min.js	200	script	(index)	38.9 KB	252 ms	
bootstrap.min.js	200	script	(index)	36.0 KB	69 ms	
jquery.cookiebar.js	200	script	(index)	8.4 KB	66 ms	
jquery.cookie.js	200	script	(index)	3.5 KB	83 ms	
popup.js	200	script	(index)	434 B	71 ms	
lightslider.css	200	stylesheet	(index)	8.2 KB	34 ms	
jquery.cookiebar.css	200	stylesheet	(index)	1.5 KB	65 ms	
font-awesome.min.css	200	stylesheet	(index)	8.1 KB	83 ms	

28 requests | 934 KB transferred | Finish: 1.13 s | DOMContentLoaded: 619 ms | Load: 1.14 s

Příloha X



Name	Status	Type	Initiator	Size	Time	Waterfall
fontawesome-webfont.woff2?v=...	200	font	(index)	75.8 KB	160 ms	
Aniuk-Bold.woff	200	font	(index)	51.0 KB	1.10 s	
Aniuk-Medium.woff	200	font	(index)	50.5 KB	1.07 s	
Aniuk-Regular.woff	200	font	(index)	46.3 KB	1.00 s	
header_hp-bg.jpg	200	jpeg	(index)	115 KB	1.53 s	
byk-plemene-charolais_item.jpg	200	jpeg	(index)	39.7 KB	527 ms	
byk-plemene-charolais-1_item.jpg	200	jpeg	(index)	32.4 KB	434 ms	
byk-plemene-charolais-2_item.jpg	200	jpeg	(index)	29.8 KB	398 ms	
byk-plemene-charolais-3_item.jpg	200	jpeg	(index)	32.5 KB	435 ms	
jalovice-plemene-charolais_item.j...	200	jpeg	(index)	30.6 KB	409 ms	
jalovice-plemene-charolais-1_ite...	200	jpeg	(index)	28.2 KB	385 ms	
burska-koza_item.jpg	200	jpeg	(index)	32.5 KB	436 ms	
burska-koza-1_item.jpg	200	jpeg	(index)	25.5 KB	346 ms	
burska-koza-2_item.jpg	200	jpeg	(index)	32.1 KB	432 ms	
bursky-kozel_item.jpg	200	jpeg	(index)	25.2 KB	336 ms	
bursky-kozel-1_item.jpg	200	jpeg	(index)	38.7 KB	522 ms	
bursky-kozel-2_item.jpg	200	jpeg	(index)	27.5 KB	373 ms	
jquery-1.12.4.min.js	200	script	(index)	38.9 KB	130 ms	
bootstrap.min.js	200	script	(index)	36.0 KB	489 ms	
jquery.cookiebar.js	200	script	(index)	8.6 KB	122 ms	
jquery.cookie.js	200	script	(index)	3.5 KB	114 ms	
popup.js	200	script	(index)	475 B	111 ms	
browser-sync-client.js?v=2.23.6	200	script	(index...	32.4 KB	436 ms	
jquery.cookiebar.css	200	stylesheet	(index)	1.6 KB	118 ms	
font-awesome.min.css	200	stylesheet	(index)	8.1 KB	106 ms	
style.css	200	stylesheet	(index)	92.3 KB	1.24 s	

34 requests | 963 KB transferred | Finish: 3.60 s | DOMContentLoaded: 1.93 s | Load: 3.50 s

Příloha XI

```
public static function download($key) {
    $data = DB::getInstance()->Select("document", ["file" => $key]);
    if ($data and isset($data['id_document'])) {
        $doc = new Document($data['id_document'],
$data['id_category'], $data['title'], $data['file_name'],
$data['file'], $data['type'], $data['position'], $data['date_add'],
$data['date_upd']);
        if (is_file(self::UPLOAD_DIR . $doc->getFile())) {
            $size = filesize(self::UPLOAD_DIR . $doc->getFile());
            header('Content-Description: File Transfer');
            header('Content-Type: application/octet-stream');
            header('Content-Disposition: attachment; filename=' .
$doc->getFileName());
            header('Content-Transfer-Encoding: binary');
            header('Connection: Keep-Alive');
            header('Expires: 0');
            header('Cache-Control: must-revalidate, post-check=0,
pre-check=0');
            header('Pragma: public');
            header('Content-Length: ' . $size);
            ob_clean();
            flush();
            readfile(self::UPLOAD_DIR . $doc->getFile());
            die();
        } else {
            return false;
        }
    } else
        return false;
}
```