Střední průmyslová škola,
Česká Lípa, Havlíčkova 426, příspěvková organizace

tel.: **487 833 123**
fax: **487 833 101**
email: **sps@sps-cl.cz**
web: **www.sps-cl.cz**

ROČNÍKOVÁ PRÁCE

VÝVOJ WEBOVÉ APLIKACE

Studijní obor: **18-20-M/01 INFORMAČNÍ TECHNOLOGIE**

Autor:

Jan Bittner

Podpis:

Vedoucí práce:

Bc. Jan Fojtík

Třída: **3.D**

Školní rok: **2014/2015**

„Prohlašuji, že jsem tuto práci vypracoval samostatně a použil jsem literárních a dalších pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.“

V České Lípě, 28. 5. 2015 Jan Bittner

...…….....

Anotace

Tato práce byla napsána za účelem poskytnutí informací veřejnosti, jak chytře tvořit moderní webové aplikace.

Obsahem práce je textová část, která obsahuje informace, které jazyky jsou vhodné k použití pro tvorbu moderních webových aplikací, základní informace a principy jazyků a popis mnou vybrané praktické části, kde je úkolem vytvořit jednoduchou webovou aplikaci, přičemž byla vybrána aplikace, redakční systém, pro blog.

Annotation

This project was written in order to grant information about creating a modern web application to the public.

It consists of a text part, which contains information about choosing proper languages for the development of a modern web application, basic information about those languages and a description of the practical part, which I have chosen. In the practical part I should create a simple easy web application (Content Management System designed for blog).

Obsah

Použité značky, zkratky a symboly

Ajax – Asynchronous JavaScript and XML

back end – část webu, která sloužící k administraci, generuje obsah pro front end

BEM – Block, Element, Modifier – styl zápisu OOCSS tříd

BFU – Běžný Fyzický (Franta) Uživatel

CSS – Cascading Style Sheets

framework – specializovaný balík knihoven pro usnadnění práce

front end – část webu, kterou vidí návštěvník webu

HTML - HyperText Markup Language

IDE – Integrated Development Environment – Vývojové prostředí

jQuery – JavaScriptová knihovna

JS – JavaScript

Less – CSS preprocesor

Minifikace – zmenšení zdrojových kódu (odstranění mezer, zalomení aj.)

MVC – Model-view-controller

MVP – Model-view-presenter

OOCSS – objektově orientované styly

PHP – PHP: Hypertext Preprocessor, původně Personal Home Page

Sass – Syntactically Awesome Style Sheets – CSS preprocesor

web – webová stránka

webový prohlížeč – internetový prohlížeč; program, kterým zobrazujeme web

1. Úvod

Webové technologie se rok co rok vyvíjí a už dlouho nám pro vývoj webové stránky, nebo aplikace, nestačí dva až tři jazyky/technologie, ale potřebujeme rovnou celý arsenál. Projekty se verzují, automatizují a stále více se také využívají frameworky v podstatě na všechno, jelikož ohromně usnadňují další vývoj – nebo by alespoň měly.

Vyznat se v současných trendech a ovládat, nebo se alespoň orientovat, dané technologie je tedy velmi náročné. V současné době se také stále více vyvíjí na bázi open-source, kde desítky, stovky, a dokonce i tisíce vývojářů pracují na jednom projektu a společně vytvářejí nádherné, a současně náročné, aplikace či technologie. Zde se na projektech uplatní ve velké míře výhody verzovacích systémů.

Jelikož je nutností používat nespočet technologií a jazyků, na psaní webové stránky si nevystačíme s poznámkovým blokem, ale pro psaní kvalitního kódu, kde nám bude dopomáhat minimálně chytré napovídání a zvýrazňování specifických prvků daného jazyka/technologie, potřebujeme speciální propracované programy - editory nebo IDE. Naštěstí pro psaní webových stránek či aplikací nepotřebují vývojáři stejný program, a tedy může každý využívat program podle svých preferencí.

Časem, když se projekty začaly čím dál více zvětšovat, začaly se také využívat automatizační nástroje, které dokáží samy kompilovat zdrojový kód, modifikovat (např. minifikace), interagovat s konzolí aj.

V mé ročníkové práci se budu zabývat vývojem webových aplikací za použití moderních technologií. Vyberu mnou používané či doporučované technologie, kterým budu věnovat samostatné podkapitoly, kde danou technologii v základu vysvětlím.

V praktické části ročníkové práce vytvořím jednoduchou webovou aplikaci, zaměřenou na redakční systém. Systém bude umět registraci a přihlášení uživatelů, spravovat články a uživatelské účty a bude obsahovat speciální administrátorské rozhraní.

1.1. Webová stránka vs. webová aplikace

V této ročníkové práci mám za úkol vytvořit webovou aplikaci, ale přitom jsem již několikrát zmínil webovou stránku. Rozdíl, mezi těmito dvěma pojmy je zároveň zásadní a současně minimální. Záleží totiž na úhlu pohledu.

Pokud se bavíme o webové stránce, je většinou myšlen front end, tedy to, co vidíme u nás ve webovém prohlížeči. Samotné vytváření stránky, resp. její generování vůbec nebereme v úvahu.

Pokud se bavíme o webové aplikaci, je většinou myšlen právě back end, tedy ta část, při které webovou stránku vytváříme. Může se jednat o nejrůznější redakční systémy, hry apod.

Mnoho lidí však tyto pojmy zaměňuje, jelikož si jsou velmi podobné a blízké – a ne vždy je to špatně. Pokud se zaměříme na statické stránky, tedy stránky, které nejsou vygenerovány žádným systémem, ale jsou napsány „ručně“, můžeme je označovat za webovou aplikaci např. v případě, kdy mají rozvinutou front end logiku a většinou tedy umí něco více, než jen vykreslit stránku – umí např. rozpohybovat stránku, vytvářet animace/prezentace, vykreslovat obrazce, obsahují hru či aplikaci aj.

Podle mého názoru se dá webová stránka považovat i za aplikaci v případech, kdy obsahuje back end nebo pokročilejší front end logiku. Pokud se chceme těmto nesrovnalostem vyvarovat, můžeme například použít univerzální zkratku *web*.

1.2. Editor vs. IDE

V úvodu ročníkové práce jsem se zmiňoval o tom, že nám pro psaní webů nestačí jednoduchý poznámkový blok, ale je nutné využívat speciální editory nebo IDE. [1], [6]

Jaký je ale mezi těmito dvěma druhy programů rozdíl? V obou případech můžeme editovat/psát naše weby s elegancí – máme hezky zvýrazněný kód a nějaké napovídání. IDE však kódu opravdu rozumí. Chápe to, co píšeme a vytváří spojení mezi částmi souborů, knihovnami, frameworky atp.

V praxi to znamená, že pokud požadujeme po editoru nápovědu v místě, kde očekáváme proměnnou, dostaneme seznam funkcí, proměnných, často se vyskytujících slov apod. Pokud budeme požadovat nápovědu na místě, kde očekáváme proměnnou, po IDE, dostaneme na výběr pouze proměnné, ke kterým si navíc můžeme zobrazit v nějakém rychlém náhledu také dokumentaci.

IDE obsahuje také spoustu nástrojů, jako například nástroje pro kompilaci, debugování (nástroj, kterým můžeme zjistit příčinu chyby) apod. Tyto nástroje se také do většiny chytrých editorů dají doplnit v podobě rozšíření, většinou vytvářené komunitou.

2. Webové prohlížeče

Pokud vyvíjíme jakýkoliv web, je nutné jej testovat v několika prohlížečích. Každý prohlížeč totiž obsahuje rozdílné jádro (systém, který vykresluje stránku), které se mnohdy liší. Každý prohlížeč také obsahuje vlastní základní styly, které je nutno resetovat pomocí speciálních stylů. Někdy se využívá také anglický, resp. mezinárodní, výraz pro webový prohlížeč – browser.

Každý prohlížeč má z pravidla svou desktopovou (stolní počítače, notebooky atp.), tak i mobilní (chytré telefony, tablety atp.) verzi. Všechny prohlížeče budu hodnotit podle procent používanosti z multiplatformního hlediska. [19], [20]

2.1. Google Chrome

Nejpoužívanější prohlížeč současnosti, u nás i ve světě, vyvíjený nad open source projektem Chromium. Prohlížeč má minimalistický design a má integrovaný Google účet. Obsahuje také vlastní centrum aplikací a doplňků. Používá ho více než 40% uživatelů. [19]

2.2. Internet Explorer

Prohlížeč vyvíjený společností Microsoft. Druhý nejpoužívanější prohlížeč na světě (cca. 15%) a třetí nejpoužívanější v Česku (cca. 19%). Starší verze byly známy pro svou zastaralost a obecně přicházely s novými technologiemi jako poslední, obvykle i s mírně upravenými. V posledních verzích se začíná situace zlepšovat, prohlížeč má dokonce nové technologie i dříve, nežli konkurence. Prohlížeč má také mobilní verzi, která se prakticky nepoužívá. [19], [20]

2.3. Mozilla Firefox

Open source multiplatformní prohlížeč. V Česku velmi rozšířený (cca. 26%), ve světě využívaný trochu méně (cca 12%). Prohlížeč má dobrou podporu doplňků, na které má vlastní centrum. [19], [20]

2.4. Další prohlížeče

Další velmi používané prohlížeče, z hlediska světového, jsou např. Safari (cca. 14%), Android (cca. 7%) a Opera (cca. 5%). [19], [20]

2.5. Desktop aplikace

Poslední dobou se také začaly budovat aplikace na webových základech. Využívá se většinou platformy Node.js a aplikace jsou postavené na základech Chromium nebo jádře webkit. Příkladem je např. editor Atom.

Díky jednoduchým webovým technologiím HTML, CSS (a jejich preprocesorů), JavaScript atp. je snadné vytvářet pro tyto aplikace mnoho doplňků. Bohužel je celková náročnost těchto aplikací značně náročnější, nežli jejich konkurenti napsané např. v jazyce C++.

3. Výběr moderních webových technologií

3.1. Kritéria výběru

Webové technologie budu vybírat podle oblíbenosti širší veřejnosti a schopnosti technologie uplatit se při vývoji webového projektu. Od každé kategorie vyberu minimálně dvě technologie, které následně zhodnotím a doporučím vhodné technologie. Obecně však platí, že na používané technologii moc nezáleží, hlavní je, že jste si nějakou konečně vybrali.

Každá technologie se dá využít, ale ne každá je dobrá právě na moderní webový projekt. Webové projekty jsou velmi specifické v tom, že se strašně rychle vyvíjí nové technologie, a jiné zase zanikají. Hodně často se také stává, že je jedna technologie na vrcholu oblíbenosti a za pár měsíců už může propadnout do zapomnění.

Nejúspěšnějšími technologiemi jsou v současnosti technologie s open source licencemi, nejčastěji např. licence MIT. Open source projekty jsou úspěšné z toho důvodu, jelikož se může zapojit v podstatě každý – a to zadarmo, což může značně pomoc v rychlosti vývoje projektu. Připojením nových vývojářů může také nastat moment, kdy někdo vnese do projektu zajímavou myšlenku a z původně nezáživného projektu se vyklube revoluční projekt, který se stane hitem.

3.1.1. Rozdělení do skupin

Můj výběr rozdělím na čtyři základní skupiny. Front end, back end, verzovací systémy a automatizační systémy. Je dobré vědět, že BFU vnímá v podstatě jenom první skupinu – front end. O ostatních vědí jen vývojáři, kteří na projektu pracují. BFU totiž neví, že se web musí nějakým způsobem vygenerovat, ale myslí si, že se mu prostě „nějak“ vykreslil do webového prohlížeče.

3.2. Front end

Do front end technologií patří všechny technologie, které nějakým způsobem zasahují do webové stránky na straně webového prohlížeče.

Nejčastěji se rozdělují do skupin jazyků, které stránku sestavují – jazyky HTML, XML, XHTML atd. –, stylují – CSS – a které s ní interagují – JavaScript, Dart atp.

Navíc, aby toho nebylo málo, mají všechny jazyky další nadstavby, které se buď do daného jazyka kompilují – preprocesory –, nebo schopnosti jazyka rozšiřují – knihovny a frameworky.

3.2.1. Sestavování

Pro sestavování webu je výběr v podstatě jistý. XHTML se již, kromě vzácných výjimek, nepoužívá. XML se v současnosti, spíše než pro tvorbu webů, používá pro uchovávání dat (jako alternativa např. k formátům JSON či CSV). Jasnou volbou je tedy jazyk HTML, který v současné nejnovější verzi HTML5 obsahuje navíc také nové sémantické elementy, podporu pro videa a zvuk atd.

Osobně neznám nikoho, a ani jsem nikde neviděl žádnou stránku, kdo by nyní používal jiný jazyk než HTML na vytváření struktury webů.

Výhody HTML jsou ty, že je značně přizpůsobivý. Nemusíme psát ukončovací značky, uvozovky u atributů, dokonce ani nemusíme elementy správně zanořovat a webový prohlížeč to „nějak“ vyřeší. A většinou dobře.

Někomu se zdá HTML přese všechno složitý, a proto existují pro HTML preprocesory, které jazyk ještě o něco ulehčují. Neznámějším HTML preprocesorem je HAML (který se také používá jako alternativa k šablonovacímu systému v Ruby on Rails, nejpoužívanějšího webového Ruby frameworku). Mnoho lidí však, místo používání HTML preprocesoru, používá nástroje, které umí HTML kód generovat z předpisu podobnému CSS selektorům a vygenerují se ihned po zapsání. Takový nástroj je např. Emmet, který používám i já.

3.2.2. Stylování

Pro stylování sestavené stránky pomocí HTML máme jen jedinou možnost a tou je jazyk CSS. Tzv. kaskádové styly žádnou jinou alternativu nemají, nebo jsem ji minimálně nenalezl.

CSS je velmi jednoduchý jazyk, který má ale tendenci být pořádně nepořádný – hlavně ve větších projektech. Platí jednoduché pravidlo – čím méně CSS, tím lépe. CSS kód je dobré rozdělovat do malých komponent tak, aby zůstal krásně přehledný.

K tomuto účelu, a nejen k němu, se využívá CSS preprocesorů, které obsahují možnost importovat jednotlivé části kódu (tzv. partials) a po kompilaci vytvoří jeden výsledný soubor se styly.

3.2.3. CSS preprocesory

Nejpoužívanějšími preprocesory jsou Sass a Less. Oba dva preprocesory dodávají do CSS možnost proměnných, cyklů, funkcí, mixinů, matematiky, vnořování aj. Jsou to tedy ideální nástroje pro tvorbu webů, hlavně pro větší projekty. [2]

Preprocesor Less se svou speciální syntaxí drží spíše blíže CSS, kdežto Sass se svou původní syntaxí dosti rozcházel od CSS, například vůbec nevyužíval složené závorky. Jeho nová syntaxe, ta se nazývá SCSS, se již podobá syntaxi CSS a je současně používána majoritně.

Oba dva preprocesory nabízí mnoho a je tak v podstatě jedno, jestli si vyberete ten nebo onen. Hlavní je si nějaký vybrat. Dobrou volbou je řídit se podle výběru spolupracovníků. Já osobně doporučuji preprocesor Sass, který se mi líbí více.

3.2.4. Interakce

Pro interakci s webem se dají využít 2 jazyky. JavaScript a Dart. Jazyk JavaScript je poměrně známý a pro weby je dostupný skoro od začátku. Kvůli jeho zvláštnímu stylu zápisu objektového kódu však vzniká mnoho různých preprocesorů, které se snaží zjednodušit syntaxi a zavést standardní objektový návrh tak, jak známe např. z jazyků jako Java, C#, C++ aj.

3.2.5. Dart

Jazyk Dart je nová technologie od Googlu, která se však v dubnu 2015 přestala vyvíjet jako náhrada JavaScriptu (do té doby byl využíván jen ve speciální verzi prohlížeče Chrome a bylo plánováno nasazení do ostatních prohlížečů) a nyní se bude vyvíjet jen jeho kompilátor *dart2js* – což znamená, že se z něho stane něco jako JavaScript preprocesor. Dart má vlastní virtuální stroj a jeho vývoj by se měl směřovat směrem na desktopy, mobily a servery.

3.2.6. Nadstavby JavaScriptu

Jeden z nejpoužívanějších preprocesorů je CoffeeScript, který velmi ulehčuje syntaxi. Píše se v něm mnoho pluginů, stránek i aplikací. Další možností je např. TypeScript, vyvíjený Microsoftem také na open source licenci.

Jednou z majoritně využívaných JavaScriptových knihoven je knihovna jQuery, která značně vylepšuje výběr/selektování konkrétních HTML elementů z webu a umožňuje s nimi dále snadnou manipulaci.

3.3. Back end

Pro back end je o mnoho více funkčních možností, jelikož se dá využít téměř jakéhokoliv jazyku, který se dá využít na serveru. Klasicky se využívají jazyky PHP, Java, VB.NET, Ruby, Python, C/C++ atd.

Nejoblíbenějším jazykem, díky velké podpoře placených, i bezplatných, serverů je jazyk PHP, který má jednoduchou syntaxi a dobrou křivku učení (je velmi jednoduchý na naučení). Běží na něm zhruba 80% všech webů. Je k dispozici mnoho frameworků, z nichž je momentálně na špici celosvětových trendů framework Laravel a pro Česko framework Nette. Všechny frameworky mají jasné základní cíle – zjednodušit syntaxi a poskytovat co možná největší úroveň zabezpečení a ochrany proti různým útokům.

3.3.1. Databáze

Do back end můžeme zařadit i databáze, kde se často používá např. Oracle, MySQL, MariaDB, SQLite aj. Databáze jsou navrženy tak, aby mohly uchovávat velké množství dat a byl k nim co nejrychlejší přístup. Nejčastěji používanou databází na menších projektech je MySQL.

3.4. Verzovací systémy

Prací na čím dál tím větších projektech není přeposílání kódu emailem či přenosnými médii zrovna ideální řešení. Nejpoužívanějšími verzovacími systémy jsou Git, Mercurial a Subversion. Tyto systémy fungují jak lokálně, tak i na sdíleném úložišti, kde se synchronizuje kód ode všech uživatelů. Jednotlivá úložiště se pak nazývají repozitáře a jsou jak placené, tak i bezplatné.

Všechny systémy mají samozřejmě své výhody a nevýhody, ale řekl bych, že Git je nejvíce rozšířený a díky službám jako je GitHub (největší úložiště bezplatných repozitářů s open source projekty) jej používá čím dál tím více lidí.

Právě na GitHubu vyvíjí tisíce vývojářů své projekty, jako např. CSS preprocesor Sass/Less, JavaScriptovou knihovnu jQuery, nový moderní editor Atom atd.

3.5. Automatizační systémy

Pokud tisíce lidí pracují na projektu, je nutné, aby všichni dodržovali stejný styl kódu, kompilovali a vyvíjeli stejným stylem atd. Zde přichází do hry nástroje na automatizaci, kde stačí jednou, v nějakém speciálním souboru, definovat co a jak se má vykonat a každý vývojář si jenom spustí daný systém a je o všechno postaráno.

Hodně využívanými systémy, minimálně v oblasti webů, jsou gulp a Grunt. Oba běží na platformě Node.js, což je JavaScriptová platforma. Grunt se konfiguruje stylem podobným způsobu JSON a gulp se konfiguruje pomocí JavaScriptu.

Sám jsem zkoušel oba dva systémy na projektu s tisíci řádky CSS, resp. Sass, kódu. V Gruntu mi kompilace trvala okolo 7,8s a v gulpu okolo 1,2s – což už je velký rozdíl.

Do automatizačních systémů můžeme zařadit také speciální soubory, které nastavují editory a IDE – jejich odsazení, kódování, přidávají na konec souboru volný řádek nebo např. mažou mezery za textem.

3.5.1. Gulp

Gulp využívá tzv. streamy, v překladu proudy, které na začátku načtou data ze souborů, s kterými mají pracovat, a na konci výsledek uloží z pravidla do jednoho souboru.

Mezi tím na datech, uložených v paměti, může být vykonána jakákoliv úloha – nejčastěji např. minifikace, kompilace CSS preprocesoru atp. Grunt se, oproti tomu, po každé úloze uloží, což zpomaluje celý proces.

Gulp obsahuje mnoho doplňků, které se dají stáhnout z balíčkového manažeru *npm*, který slouží pro platformu Node.js.

4. Popis vybraných technologií

V následujících kapitolách popíšu základy vybraných jazyků a způsoby, jak se s nimi dají efektivně vyvíjet weby. Ukážu jak úplné základy, tak i pokročilejší způsoby, které se běžně na vývoj využívají.

4.1. HTML

Z počátku je dobré uvést, že HTML není jazyk programovací, nýbrž jazyk značkovací. Skládá se tedy ze značek, kterými popisuje obsah webu. HTML nijak neřeší vzhled webu – k tomu je určen jazyk CSS, který je popsán v další kapitole.

Každá stránka se píše zvlášť do samostatného souboru s příponou *html*.

4.1.1. Elementy a značky

HTML je složen z elementů, které jsou složeny ze značek, někdy označovaných jako tagy, a obsahu mezi nimi. Elementy mohou být párové a nepárové. Nepárovým elementem je například obrázek, párovým například odstavec. Obecně platí, že nepárové elementy nemohou mít, a ani nemají, žádný obsah.

Značky se zapisují názvem elementu ohraničeným špičatými závorkami. Ukončovací značka má ještě před názvem elementu lomítko. Značky se zapisují malými písmeny.

Základním blokovým elementem – to jsou ty elementy, které vyplní šířku stránky a jsou od zbytku stránky odděleny – je element *div*.

```
1. <div></div>
```

V HTML se mohou psát i komentáře, které se píší stylem, jak je ukázáno na ukázce níže. V HTML jsou jenom blokové komentáře a neexistují řádkové komentáře.

```
1. <!-- komentář -->
2.
3. <!-- víceřádkový
4.      komentář -->
```

V HTML5 přibyly nové sémantické elementy, což jsou elementy, které mají nějaký smysl. Dobrým příkladem je např. element *header*, který se využívá pro hlavičku. Tyto elementy se v podstatě chovají jako klasický blokový element – např. *div*. Dalšími

zástupci nových sémantických elementů jsou také elementy *footer*, *article*, *sidebar*, *nav*, *main* atd.

4.1.2. Atributy

Elementy také obsahují atributy, které mohou obsahovat např. informace o tom, kde se nachází obrázek – to je konkrétně atribut *src*. Ve verzi HTML5 se také dají využívat vlastní atributy, které musí být zapsány s prefixem *data*.

Speciálními atributy jsou atribut *id* a *class*. Atribut *id* označuje unikátní element a musí být na stránce jedinečný. Atribut *class*, v překladu třída, vyznačuje skupinu elementů.

```
1.  <p class="poznamka">Sem se může napsat poznámka.</p>
2.
3.  <img src="ovoce/jablko.png" alt="Obrázek jablka." data-ovoce="jablko"/>
```

Názvy atributů i jejich hodnoty je dobré psát bez diakritiky, pokud není zapotřebí, jako např. v ukázce níže u popisku obrázku – atribut *alt*.

4.1.3. Definování hlavičky

HTML stránka se musí nastavit – kódování, titulek, autor atp. – v hlavičce webu. Celý web musí být umístěn v elementu *html*, který obsahuje element *head* (hlavička) a *body* (tělo). Element *body* obsahuje viditelný obsah webu.

```
1.  <!DOCTYPE html>
2.  <html lang="cs">
3.  <head>
4.      <meta charset="UTF-8" />
5.      <title>Moje úžasná stránka</title>
6.      <meta name="description" content="Popisek mého skvělého webu." />
7.      <meta name="author" content="Jan Bittner" />
8.  </head>
9.  <body>
10.
11.   <p>Odstavec textu.</p>
12.
13. </body>
14. </html>
```

Samotný *html* element obsahuje také atribut, který nastavuje jazyk webu. Před elementem *html* se nastavuje tzv. *doctype*, který deklaruje typ dokumentu.

Do HTML hlavičky se píší meta tagy, které nastavují web. Nejčastěji se nastavuje kódování stránky – aby prohlížeč věděl, v jakém je nutné web zobrazit –, popis webu,

autor webu, klíčová slova atp. Nutné je zadat pouze kódování. Meta tagy mohou sloužit jako zdroj informací pro vyhledávače.

Kromě meta tagů se v hlavičce deklaruje také titulek stránky, který BFU, resp. návštěvník webu, uvidí jako popisek záložky prohlížeče. Na titulek stránky se využívá element title.

4.2. CSS

Pokud chceme web stylovat, tedy měnit vzhled, musíme použít jazyk CSS. Styly se píší do samostatného souboru. Z pravidla, kvůli optimalizaci, využíváme pouze jednoho výsledného CSS souborů. Soubor zapisujeme, nejčastěji s názvem styly (nebo anglicky style), s příponou *css*.

Soubor se styly musíme se souborem HTML propojit právě v HTML hlavičce, nikoliv však pomocí meta tagu, ale pomocí elementu link. Jak takové propojení vypadá, je ukázáno v kódu níže.

```
1.  <link rel="stylesheet" href="css/style.css" type="text/css"/>
```

Stejně jako HTML, ani CSS není programovacím jazykem.

4.2.1. Selektory

Jazyk CSS staví na jednoduchém principu. Nejdříve určíte elementy, které požadujete – tomu se říká selektor – a následně pro selektor vytvoříte blok, kam zapisujete, stylem *vlastnost: hodnota*, jednotlivé styly.

Jako selektor se může využít název elementu, atribut elementu, *třída*, *id* atd. Může se také vytvářet tzv. multi selektor, který spojuje více selektorů pro jeden blok stylů.

4.3. Preprocesor Sass

Hlavně ve větších webových projektech se časem začala projevovat nepořádnost CSS, a proto se začaly vyvíjet CSS preprocesory, které kromě možnosti rozdělit kód nabízí také prvky z programovacích jazyků. Získáme možnost proměnných, funkcí, mixinů (obdoba void funkcí – funkce, které nemají návratovou hodnotu), cyklů, zanořování, podmínek atd. [2]

Jednotlivé třídy, které se nejčastěji používají pro stylování elementů, můžeme např. generovat například podle mapy (něco jako pole) a tím udělat kód méně redundantní (tzn. méně zbytečného a nadbytečného kódu). [3]

```
1. @for $i from 1 through 3 {
2.   .#{sloupec}-$i {
3.     width: 100px * $i;
4.   }
5. }
```

Ukázka kódu výše vygeneruje třídu *sloupec-1*, *sloupec-2* a *sloupec-3*, každou s jinou šířkou elementu. [4]

4.3.1. Kompilace z preprocesoru

CSS preprocesor se musí kompilovat na CSS kód. To z toho důvodu, jelikož webové prohlížeče neumí preprocesorové styly zobrazit, se musí vytvořit s každou změnou nový výstupní CSS soubor.

Sass má dva hlavní kompilátory – Ruby a LibSass. Hlavní kompilátor je ten v jazyce Ruby, ale jazyk Ruby je poměrně pomalý, a proto je dostupná také druhá možnost, a to využití LibSass. LibSass je psán v jazycích C a C++ a je tedy mnohem rychlejší.

Samotnou kompilaci můžete také obstarávat např. pomocí Node.js doplňků, které se za vás budou obstarávat kompilaci, a vy se o nic nemusíte starat.

4.3.2. Princip objektově orientovaných stylů

OOCSS je postaveno na myšlence, že máme web složen z několika komponent, které mohou mít drobné rozdíly. S touto metodou tedy píšeme styly tak, že vytváříme komponenty. Komponenta by měla být nezávislá na zvoleném elementu a svém umístěním v kódu, což zaručí nízkou specifičnost selektorů, což je jen dobře. Každá komponenta by měla fungovat stejně dobře ať je použita jak v hlavičce, tak třeba v patičce.

Komponentě můžeme také říkat objekt. Každý objekt může mít i své potomky a objekt i potomek mohou mít modifikátory. Modifikátory se užívají v případech, kdy je nutná jen minimální změna stylů a tudíž nemá smysl vytvářet novou komponentu.

Potýkáme se také se systémem, jakým způsobem pojmenovávat třídy tak, abychom mohli jasně, a přehledně, zapsat objekt, potomka, i modifikátory. Jedním z řešení je použití metody BEM.

Objekt zapíšeme klasicky, jako normální třídu. Potomka zapíšeme jako složeninu názvu objektu a názvu potomka oddělených dvěma podtržítky. Modifikátory poté jednoduše zapíšeme oddělením dvěma pomlčkami.

```
1.  .clovek {}
2.
3.  .clovek__ruka {}
4.
5.  .clovek__ruka--prava {}
```

Na ukázce výše můžeme vidět styl zápisu BEM aplikovaného na příklad pravé ruky člověka. Pokud bychom chtěli využít jedné z výhod Sass – zanořování –, kód by vypadal takto. Znak & zastupuje nadřazený selektor.

```
1.  .clovek {
2.
3.    &__ruka {
4.
5.      &--prava {}
6.    }
7.  }
```

4.4. JavaScript

Jazyk JavaScript je nedílnou součástí webů. Je to poměrně jednoduchý skriptovací (to znamená, že pro svůj chod se nemusí kompilovat, ale je interpretován) jazyk, který se využívá na logiku front endu. Nejedná se však o jazyk Java, který se částečně shoduje jen v názvu, jinak jsou velmi rozdílné.

Pomocí JavaScriptu můžeme na stránce měnit obsah, přidávat/odebírat elementy, přesouvat se plynule po stránce atp. Jazyk běží na straně klienta, konkrétně v prohlížeči. Pomocí jednoduchého scriptu můžeme do stránky přidat nějaký text, resp. s ním stránku přepsat.

```
1.  document.write("Nový text na stránce.");
```

V JavaScriptu můžeme využívat také např. proměnné, které se definují pomocí klíčového slova *var*. Pokud klíčové slovo *var* při deklaraci proměnné nezadáme, stává se proměnná globální, tedy dostupnou v celém programu. Můžeme však využívat také

podmínky, switche, ternární operátory, cykly, funkce atd. Zajímavé na tomto jazyku je, že do proměnných můžeme uložit také funkce.

```
1.  var a = 40;
2.  var b = "pes";
3.  c = "auto";
```

V JavaScriptu můžeme také jednoduše vypsat hlášku – kde se využívá vyskakovacího okénka. Využívá se pro to funkce *alert*.

```
1.  var chyba = function(hlaska) {
2.    alert(hlaska);
3.  }
4.
5.  chyba("Nastala nějaká chyba.");
```

Pokud budeme chtít pracovat např. s HTML elementem, který má nějaký atribut *id* a změnit mu obsah elementu, můžeme to provést ukázkovým kódem níže. Nejdříve si do proměnné uložíme element, který získáme podle ID *tlacitko*. Následně pro tento element změníme textový obsah.

```
1.  var element = document.getElementById("tlacitko");
2.  element.textContent = "Klikni na mě!";
```

4.4.1. Knihovna jQuery

Knihovna jQuery zjednodušuje výběr elementů, který můžeme napsat stejným způsobem, jako selektory v CSS. Kromě zjednodušené možnosti výběru elementu, či elementů, knihovna poskytuje zjednodušené ovládání např. pro změnu textu/HTML v elementu, animace atp.

```
1.  $('.clanek p').text('Změnil jsem text článku.');
```

V ukázce výše byly vybrány všechny elementy *p*, které jsou potomky elementu s třídou *clanek*. Pomocí funkce *text* byl následně změněn textový obsah těchto elementů. Pokud bychom napsali HTML kód, zapsal by se jako text. [5]

4.5. PHP

Jazyk PHP je nejběžnější jazyk, který se používá pro vývoj webu na straně back endu. V tomto jazyku píšeme kód, který nám následně vygeneruje HTML kód podle nějakých parametrů – většinou podle url adresy webu.

Syntaxe se podobá běžným jazykům, jako například jazyku C nebo Java. Jazyk je hojně využíván začátečníky, právě díky jeho jednoduché syntaxi a velké komunitě, na kterou se mohou, na nejrůznějších diskuzních fórech, obrátit s pomocí.

PHP je sestavován na serveru, který musíme mít i pro lokální vývoj webu. PHP aplikace, jak jsem již nastínil výše, reaguje vždy na požadavek, kterému se pokusí vyhovět a vrátí výstup, např. v podobě HTML. To také znamená, že se nedá s aplikací interagovat jinak, než pomocí požadavků, které se odešlou pouze při obnově záložky prohlížeče – tzv. refresh, nebo při příchodu na web.

Samozřejmě existují způsoby, jak tento problém částečně odstínit. Jednou z možností může být technologie Ajax, která posílá speciální požadavky i bez refreshe webu, a to například pomocí funkce dostupné v JavaScriptové knihovně jQuery.

4.5.1. Základní konstrukce jazyka

Na začátek každého souboru musíme vložit speciální značku, která říká, že začíná PHP kód. Značka, včetně ukončovací, která není povinná a ukončí se automaticky koncem souboru, je ukázána na ukázce níže.

```
1.  <?php
2.
3.  ?>
```

Základním příkazem jazyka je funkce *echo*, která vypíše obsah argumentu funkce do webu. V PHP můžeme využívat proměnné, které se píší s prefixem dolaru – znak $. Pro proměnné se neurčuje datový typ, jelikož je jazyk dynamicky typovaný, stejně jako JavaScript.

```
1.  <?php
2.
3.  $a = 5;
4.  $b = "Ahoj světe!";
```

Oproti jiným programovacím jazykům spojování textových řetězců neprovádíme znakem plus, ale tečkou.

```
1.  <?php
2.
3.  $a = 1;
4.  $b = 2;
5.
6.  echo($a + $b); // Výstup 3
7.  echo($a . $b); // Výstup 12
```

V PHP se rozlišuje mezi textovým řetězcem napsaným v uvozovkách a v apostrofech. Pokud do textového řetězce zapsaného v uvozovkách vložíme název proměnné, proměnná se vypíše.

Dále můžeme využívat v PHP klasických funkcí jako podmínky, cykly, pole, seznamy atp. V PHP se dá psát objektově orientovaný kód, a tedy můžeme využívat i třídy. Příklad objektově orientovaného kódu je ukázán na ukázce níže, ve které je třída člověka, který má metodu pro chození a promluvení.

```
1.  <?php
2.
3.  class Clovek {
4.
5.    public function krok() {}
6.
7.    public function promluv($zprava) {}
8.  }
```

4.5.2. Architektura MVC

Pokud chceme vytvářet náročnější webové aplikace, je nutné, abychom se drželi nějaké architektury, která vytváří pomyslnou strukturu aplikace a zjednodušuje její chod.

Jednou z nejpoužívanějších metod je metoda MVC nebo MVP. Tyto dvě metody si jsou velmi podobné a někdy jsou i zaměňovány. Pro jednoduchost budeme nazývat tuto metodu architekturou MVC.

Princip spočívá v tom, že oddělujeme logiku a výstup. Aplikaci rozdělujeme do tří typů – Model, View (pohled), Controller (kontroler). Modely obstarávají celou logiku aplikace a může to být například odesílání emailu, správce uživatelů atp., pohledy jsou zpravidla HTML šablony, které jsou obohaceny o jednoduchý PHP kód a cykly a kontrolery vytvářejí propojení mezi modely a pohledy a řeší co vykonat na základě daných parametrů pro sestavení stránky – které se získají např. z url adresy webu.

Uživatel stránky zadá url adresu. Adresu zjistí kontroler, který ji zpracuje a rozloží na jednotlivé části. Pokud kontroler rozhodne, že je adresa platná, zavolá příslušný model, který vykoná své funkce a vrátí data kontroleru. Kontroler zavolá pohled, kterému dodá data, které se pomocí cyklů atp. vypíší. Web se nechá vykreslit a uživatel vidí novou stránku.

4.6. Git

Git je verzovací systém, který se stará o verzování projektu, rozklad projektu na více částí a případné řešení nesrovnalostí, resp. kolizí, pokud např. dva lidé vloží do stejné části souboru rozdílný kód.

Git si stáhneme z webových stránek projektu a po úspěšné instalaci dostaneme možnost jej používat v konzoli. K dispozici je také řada grafických programů, mezi nimiž je oblíbený např. ten od společnosti GitHub Inc. V ročníkové práci budu ukazovat příkazy používané v konzoli, kterou na svých projektech používám.

Každá uložená skupina změn, ta obsahuje jednotlivé dílčí změny v souborech, se nazývá commit.

4.6.1. Řešení kolizí

Pokud nastane problém, kdy dvě verze kolidují, mohou nastat 2 případy. V první případě, tom lepším, systém sám rozdíl detekuje a správně výsledný soubor uloží. V druhém případě, pokud si systém nebude vědět rady, která verze je ta správná, vypíše do konzole hlášku. Z hlášky je vždy patrné, jaký soubor, či soubory, kolidují a je pak jen na uživateli, aby kolizi ručně upravil.

4.6.2. Základní příkazy

Git obsahuje mnoho příkazů, které může uživatel používat, avšak pro základní práci jich uživatel potřebuje znát jen několik.

Příkaz *git status* do konzole vypíše aktuální informace o změnách v souborech. Vypíše, které soubory byly změněny, které smazány a které byly uloženy pro budoucí commit.

Příkaz *git pull* nahraje nejnovější verzi ze sdíleného repozitáře. Tento příkaz také musíme zadat vždy před tím, než odešleme data na sdílený repozitář.

Příkaz *git add --all* uloží všechny změněné soubory pro budoucí commit. Namísto *--all* můžeme také zadat cestu k chtěnému souboru.

Příkaz *git commit -m "Popis commitu."* uloží dané soubory do lokálního repozitáře s příslušným popisem.

Příkaz *git push* nahraje změny z lokálního repozitáře do repozitáře sdíleného. Před použitím tohoto příkazu je nutné zadat příkaz *git pull*.

4.6.3. Vytvoření sdíleného repozitáře

Vytvoření sdíleného repozitáře se provádí na speciálních webových aplikacích, které většinou poskytují veřejné repozitáře zdarma. Nejznámější a zároveň nejpoužívanější je úložiště GitHub. Na tomto úložišti si může každý uživatel vytvořit neomezeně mnoho veřejných repozitářů, za soukromé se však již platí.

Pro získání dat z repozitáře se musí tzv. naklonovat. Příkaz, kterým repozitář naklonujete do svého počítače, najdete na pravé straně webu. Na klonování se využívá příkazu *git clone*.

4.7. Automatizační systém gulp

Pro automatizaci projektu doporučuji využívat systém gulp, který funguje na bázi streamů. To znamená, že na začátku procesu se jednou načtou data a na konci, klidně po nesčetném množství úprav, se jednou data uloží. Díky tomuto, jelikož práce s diskem je pomalá, je gulp velmi rychlý. Data, a všechny procesy na nich, jsou po celou dobu v paměti RAM.

4.7.1. Instalace

Jelikož gulp funguje nad platformou Node.js, je zapotřebí jako první nainstalovat Node.js. V konzoli, pro instalaci doplňků atp., budeme využívat příkazu npm, který přistupuje k balíčkovému systému Node.js.

Samotná instalace gulpu poté probíhá v konzoli, kde nejdříve nainstalujeme gulp globálně pomocí příkazu *npm install --global gulp*. Přepneme se do složky projektu a jako první nainstalujeme doplněk gulp, pomocí příkazu *npm install --save-dev gulp*. [7]

Po instalaci gulpu musíme vytvořit soubor gulpfile.js, kam budeme psát naše automatizační úlohy. Do souboru vložíme následující kód. Na řádku 3 můžeme vidět princip, jak se jednotlivé úlohy zapisují. [7], [8]

```
1. var gulp = require('gulp');
2.
3. gulp.task('default', function() {});
```

4.7.2. Instalace doplňků

Instalace doplňků probíhá obdobným způsobem, jako instalace gulpu, což je také doplněk. V balíčkovém systému npm si najdeme chtěný gulp doplněk, které mají z pravidla prefix *gulp*.

Zjistíme si název doplňku, namátkou např. *gulp-sass*, a do konzole zadáme příkaz *npm install --save-dev gulp-sass*.

4.7.3. Doporučené doplňky

Pro webové projekty je vhodné použít doplněk *gulp-sass*, který se stará o kompilaci CSS preprocesoru Sass, doplněk *gulp-autoprefixer*, který přidává do výsledného CSS kódu prefixy pro dané CSS vlastnosti, doplněk *gulp-rename*, díky kterému můžeme přejmenovat výsledný soubor, doplněk *gulp-notify*, který nám např. při chybě může zobrazit informační bublinu a např. doplněk *gulp-sourcemaps*, který nám propojí CSS kód se zdrojovým Sass kódem. [8]

Je dobré najít si doplňky, které sami využijete na vašem projektu a při vašem stylu práce.

5. Popis praktické části ročníkové práce

V této kapitole popíši funkce aplikace a využité jazyky a technologie. Také nastíním možnosti, jak se systém dá do budoucna vylepšit a modifikovat.

V praktické části ročníkové práce jsem vycházel z tutoriálu pro jednoduchý redakční systém MVC od Davida Čápky, který popisuje vývoj základních funkcí, propojení systému s databází a vytvoření hezkých url adres. [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]

5.1. Funkce aplikace

Aplikace, která slouží jako správce, resp. redakční systém, menšího blogu, je plně provozu-schopná. Jakýkoliv návštěvník se může registrovat a přidávat své články. Administrátoři mohou editovat a mazat všechny články, bez výjimek, a také mohou měnit administrátorská práva ostatním uživatelům, kterým se změna ihned projeví.

Návštěvník webu si může zobrazit seznam článků, pro které je uveden titulek, popis a autor daného článku. Návštěvník může článek rozkliknout (resp. otevřít), pro další četbu, nebo rozkliknout účet autora.

Každý uživatel má svůj, veřejně přístupný, profil. V profilu se kromě informací, motta, popisku a odkazů na sociální sítě zobrazuje také výpis článků, které daný uživatel napsal. Pod uživatelovou přezdívkou se, v případě, že je uživatel také administrátorem, zobrazí příslušný titul administrátora.

Návštěvník si také může otevřít přehled všech uživatelů, kde je graficky, změnou barvy písma, odlišen každý administrátor.

5.1.1. Psaní a editace článku

Každý uživatel, který se přihlásí do administračního rozhraní, může, otevřením záložky *editor článků*, začít psát, nebo editovat, článek. Článek se do blogu přidá ihned po jeho odeslání.

Pokud chce uživatel svůj článek editovat, musí otevřít záložku *seznam článků* v administrátorském rozhraní, kde je vždy u daného článku možnost editace. Článek lze také zobrazit k editaci úpravou url adresy.

5.1.2. Úprava profilu uživatele

Uživatel může v administračním rozhraní změnit veřejné informace pro svůj profilový účet. Nastavení, obsahující formulář, lze otevřít pomocí záložky *účet*.

5.2. Využité jazyky a technologie

V praktické části ročníkové práce jsem využil jak verzovacího systému Git, tak i automatizačního systému gulp. Jako front end technologie jsem využil jazyky HTML5 a CSS3 s preprocesorem Sass.

Pro back end jsem využil databázi MySQL. Databáze obsahuje 4 tabulky. Obsahuje tabulku *articles*, která obsahuje informace o článcích, tabulku *users*, která obsahuje informace o uživatelích a také hash (otisk) hesla, tabulku *url*, která slouží pro oddělení klíčových slov použitých jako hlavní části url adresy, a tabulku *url_commands*, která obsahuje klíčová slova sloužící pro příkazy v url adrese.

Jako serverový jazyk jsem zvolil jazyk PHP, který je jednoduchý na naučení a má dostupné řešení u hostingových společností.

Využité technologie a jazyky jsou popsány v předchozí kapitole, kde jsou vysvětleny základy a principy využití a použití.

5.3. Možnosti modifikace

Tato jednoduchá aplikace, která se dá lehce modifikovat, určitě postrádá nějaké prvky, které jsou na reálném projektu nutností.

Na reálném projektu by se určitě hodilo, aby se článek nejdříve schválil, nebo aby se nejdříve schválil uživatel a byly mu povoleno psát články. Také by bylo vhodné zavést systém hodnocení článků, možnost psát komentáře atp.

Uživatelům by se mohlo vytvořit více rolí, a tak by se mohla udělat např. role moderátora, který by mohl např. schvalovat články.

Užitečným nástrojem by také mohlo být posílání soukromých zpráv, aby spolu mohli jednotliví uživatelé komunikovat přímo v systému.

Vhodnou inovací by také mohlo být vytvoření kanálů, které by sdružovali články podle nějakého klíčového elementu. Obdobný systém by se dal vytvořit za použití tagů, kterými by autor článku označil článek.

Modifikací se dá jistě vymyslet mnoho, a jelikož se dá systém lehce modifikovat, není potíž vymyslet opravdu cokoliv.

6. Závěr

V ročníkové práci jsem vytvořil jednoduchou funkční webovou aplikaci, která obstarává chod blogu. Aplikace je napsána na straně serveru a generuje jednotlivé stránky webu podle hezké url adresy, čehož je docíleno pomocí MVC architektury, která značně zjednodušuje celý proces.

Pro back end část této aplikace jsem zvolil serverový jazyk PHP, který komunikuje s databází MySQL, která shromažďuje veškerá data. Tyto jazyky jsem zvolil kvůli jejich dostupnosti a snadnému zápisu.

Pro front end část této aplikace jsem zvolil jazyky HTML5 a CSS3, se Sass preprocesorem, které jsou dnes již standardem. Preprocesor Sass aplikaci umožňuje lepší správu stylů, což je určitě výhodné.

Webová aplikace se mi podařila úspěšně vytvořit a celý systém funguje tak, jak bylo zamýšleno. Aplikace, resp. blog, je do budoucna jednoduše modifikovatelný a dá se snadno rozšířit o řadu inovací.

Během tvorby této ročníkové práce jsem se naučil jak efektivně pracovat s automatizačními systémy, zlepšil jsem svoje dovednosti v synchronizačním nástroji Git, jazyku PHP, pro který jsem se také naučil MVC architekturu a zvětšil jsem své povědomí o komplexnosti webových aplikací.

7. Použité zdroje

[1] BITTNER, Jan. Moderní a profesionální webové portfolio: Příprava a nastavení [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/html-css-moderni-portfolio-priprava-nastaveni

[2] BITTNER, Jan. Moderní a profesionální webové portfolio: Úvod do CSS preprocesoru Sass [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/tutorial-moderni-webove-portfolio-sass

[3] BITTNER, Jan. Moderní a profesionální webové portfolio: Datové typy v SASS [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/tutorial-moderni-portfolio-sass-datove-typy

[4] BITTNER, Jan. Moderní a profesionální webové portfolio: Cykly a @media v SASS [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/tutorial-moderni-portfolio-sass-cykly-a-media

[5] BITTNER, Jan. Základy jQuery: Vkládání obsahu v jQuery (DOM) [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/tutorial-jquery-dom-vkladani-obsahu

[6] BITTNER, Jan. Sublime Text [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/sublime-text

[7] Gulp: Getting Started [online]. [cit. 2015-05-20]. Dostupné z: https://github.com/gulpjs/gulp/blob/master/docs/getting-started.md

[8] Owl CSS: gulpfile.js [online]. [cit. 2015-05-20]. Dostupné z: https://github.com/owlcss/owlcss/blob/master/gulpfile.js

[9] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Popis MVC architektury [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-popis-architektury

[10] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): .htaccess, autoloader a obecný kontroler [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-htaccess-autoloader-kontroler

[11] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Směrovač (router) [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-router-smerovac

[12] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Propojení kontroleru a pohledu [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-propojeni-kontroleru-a-pohledu

[13] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Kontaktní formulář [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-kontaktni-formular

[14] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Založení databáze a přístupy k ní v PHP [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-zalozeni-databaze

[15] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Databázový wrapper [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-pdo-crud-wrapped

[16] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Výpis článků z databáze v PHP (MVC) [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-vypis-clanku

[17] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Zabezpečení šablon [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-zabezpeceni-sablon

[18] ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC): Mechanismus zpráv [online]. [cit. 2015-05-20]. Dostupné z: http://www.itnetwork.cz/objektovy-mvc-redakcni-system-php-mechanismus-zprav

[19] StatCounter Global Stats: Top 9 Browsers from Apr 2014 to Apr 2015 [online]. [cit. 2015-05-20]. Dostupné z: http://gs.statcounter.com/#all-browser-ww-monthly-201404-201504

[20] StatCounter Global Stats: Top 9 Browsers from Apr 2014 to Apr 2015 [online]. [cit. 2015-05-20]. Dostupné z: http://gs.statcounter.com/#all-browser-CZ-monthly-201404-201504

8. Seznam použitého software

[21] Jetbrains s.r.o.: *PhpStorm 8.0.3*

[22] GitHub Inc.: *Atom 0.199.0*

[23] Apache Friends: *XAMPP Control Panel v3.2.1*

[24] Microsoft Corporation: *Microsoft Office Word 2013*

[25] Alex Gorbatchev: *SyntaxHighlighter 3.0.83*

9. Seznam příloh

A. Datové DVD obsahující:

- Text práce ve formátu PDF
- Soubory s webovou aplikaci

10. Licenční ujednání

Ve smyslu zákona č. 121/2000 Sb., O právu autorském, o právech souvisejících s právem autorským, ve znění pozdějších předpisů (dále jen autorský zákon) jsou práva k maturitním nebo ročníkovým pracím následující:

Zadavatel má výhradní práva k využití práce, a to včetně komerčních účelů.

Autor práce bez svolení zadavatele nesmí využít práci ke komerčním účelům.

Škola má právo využít práci k nekomerčním a výukovým účelům i bez svolení zadavatele a autora práce.